TorqueBox

The Ruby Application Platform

3.1.2

by The TorqueBox Project

What is TorqueBox?

TorqueBox provides an enterprise-grade environment that not only provides complete Ruby-on-Rails and Rack compatibility, but also goes beyond the functionality offered in traditional Rails/Rack environments.

1. Built upon JBoss AS

Instead of building a Ruby Application Platform from the ground-up, TorqueBox leverages the existing functionality JBoss has been shipping for years in the JBoss Application Server. JBoss AS includes high-performance clustering, caching and messaging functionality. By building Ruby capabilities on top of this foundation, your Ruby applications gain more capabilities right out-of-the-box.

2. Built upon JRuby

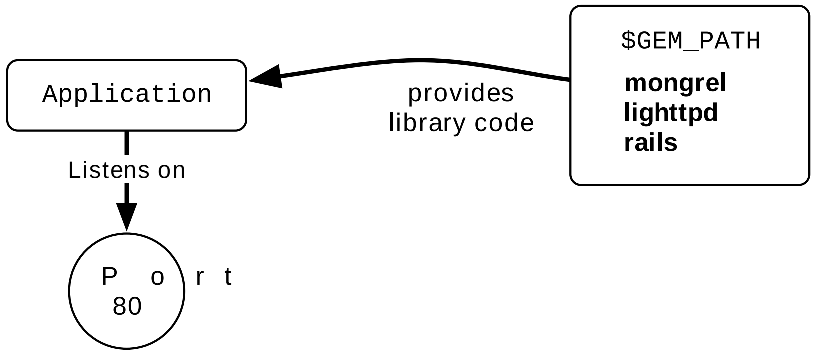
JRuby is a fast, compliant implementation of the Ruby language upon the Java Virtual Machine. Pure Ruby applications run un-modified within the JRuby interpreter. By binding JRuby to the components within JBoss, their functionality is exposed in a manner suitable to Rubyists.

3. Open-Source

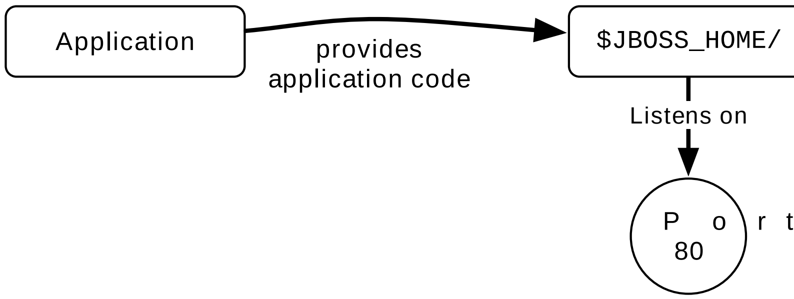
TorqueBox is a product of the JBoss Community, and is completely open-source software. TorqueBox is licensed under the LGPL. You may download the binaries or the source-code, modify it if you desire, and use it, even for profit, without any licensing costs.

4. The "application platform" concept

Traditionally, Ruby applications were responsible for their services from the ground-up. You literally ran the application. It would import support libraries to handle HTTP listening, for example.



An application platform provides the foundations for any and all application functionality. The deliverable application itself does not need to handle the networking layers, the messaging facilities or the clustering logic. This is provided to the application "for free".



TorqueBox Installation

1. Distribution Variants

Starting with TorqueBox 3, there are two release distributions available:

- slim - A smaller TorqueBox distribution that includes only the functionality needed to run Ruby applications and basic Java web applications (comparable to what would run on Apache Tomcat)
- full - The full JBoss AS distribution with TorqueBox included. Use this if you need to run JavaEE and Ruby applications in the same application server.

2. Getting Started Guide

For complete installation details and a walkthrough of creating sample applications please refer to the Getting Started Guide [http://torquebox.org/getting-started/3.1.2/].

3. Setting JVM Properties

If using the `torquebox` command (Chapter 17, The torquebox Command), JVM properties can be set with the -J flag. The only caveat is hypens must be escaped with a "\".

For example:

```
$ torquebox run -J "\-Xmx2048m \-Djruby.jit.logging=true"
```

If using `standalone.sh`, just append the JVM properties to the end of the command.

For example:

```
$ $JBOSS_HOME/bin/standalone.sh -Djruby.jit.logging=true
```

If you'd prefer not to pass the JVM properties on the commandline, they can also be set in `$JBOSS_HOME/bin/standalone.conf` by appending to the `JAVA_OPTS` variable.

For example:

```
...
if [ "x$JAVA_OPTS" = "x" ]; then
   JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -Djava.net.preferIPv4Stack=true
    -Dorg.jboss.resolver.warning=true    -Dsun.rmi.dgc.client.gcInterval=3600000    -
Dsun.rmi.dgc.server.gcInterval=3600000"
```

```
    JAVA_OPTS="$JAVA_OPTS -Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -Djava.awt.headless=true"
   JAVA_OPTS="$JAVA_OPTS -Djboss.server.default.config=standalone.xml"
else
    echo "JAVA_OPTS already set in environment; overriding default settings with values: $JAVA_OPTS"
fi

JAVA_OPTS="$JAVA_OPTS -Djruby.jit.logging=true"
```

You can also set the JAVA_OPTS environment variable directly in the shell, but if you do this make sure you copy the increased MaxPermSize from `$JBOSS_HOME/bin/standalone.conf` since if JAVA_OPTS is set in the shell the defaults in this file won't be applied.

For example:

```
$ export JAVA_OPTS="-Xmx1024m -XX:MaxPermSize=256m -Djruby.jit.logging=true"
```

4. Setting JRuby Properties

While some JRuby properties can be set via torquebox.yml or torquebox.rb as shown in Section 2.4, “Ruby runtime configuration”, JRuby supports many more options than the ones TorqueBox exposes directly. As long as you are using the `torquebox` command (Chapter 17, The torquebox Command), you can set these other properties with the `JRUBY_OPTS` environment variable.

For example:

```
$ export JRUBY_OPTS="-X+C --1.9 -Xjit.logging=true"
$ torquebox run
```

If you are not using the `torquebox` command, `JRUBY_OPTS` can still be used for options like "--1.9", "--1.8", "-X+C", "+X-O", but not for options of the style "-Xa.b" like "-Xjit.logging=true". The latter options will need to be set as JVM properties (Section 3, “Setting JVM Properties”) by prefixing them with "jruby." like "-Djruby.jit.logging=true".

For example:

```
$ export JRUBY_OPTS="-X+C --1.9"
$ $JBOSS_HOME/bin/standalone.sh -Djruby.jit.logging=true
```

JBoss AS Crash Course

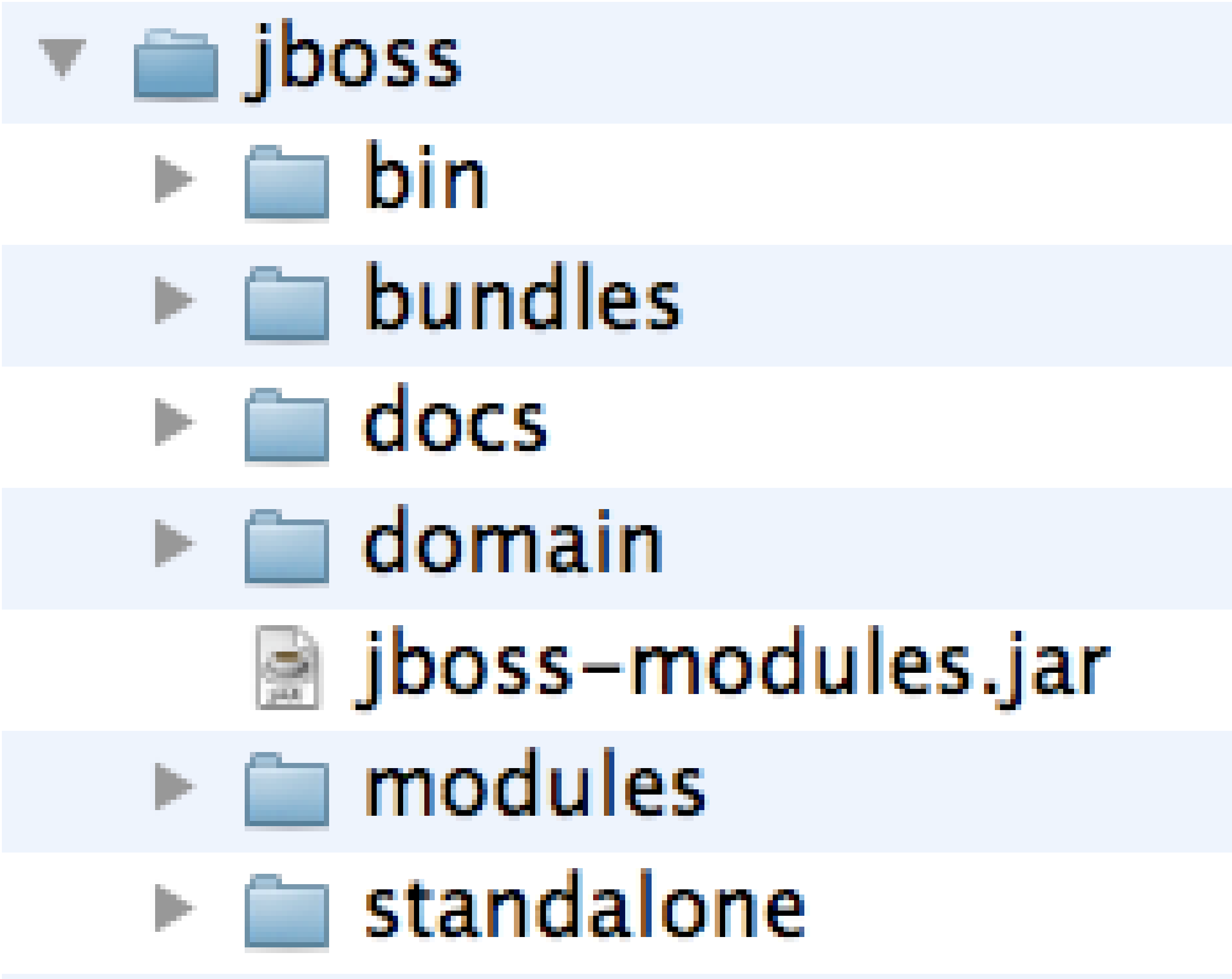
The JBoss Application Server (AS7) is the foundation upon which TorqueBox is built. You can go a long way with TorqueBox without knowing anything about the administration of JBoss AS, but for advanced applications, it's worth knowing something about how AS is configured and extended. Feel free to skip this section if you're just getting started with TorqueBox, and use it as a reference later.

For more detailed documentation, please read the official AS7 docs [https://docs.jboss.org/author/display/AS71/Documentation].

1. Configuring

JBoss AS7 has extensive changes since the previous releases, the most visible being folder structure and runtime modes: domain and standalone.

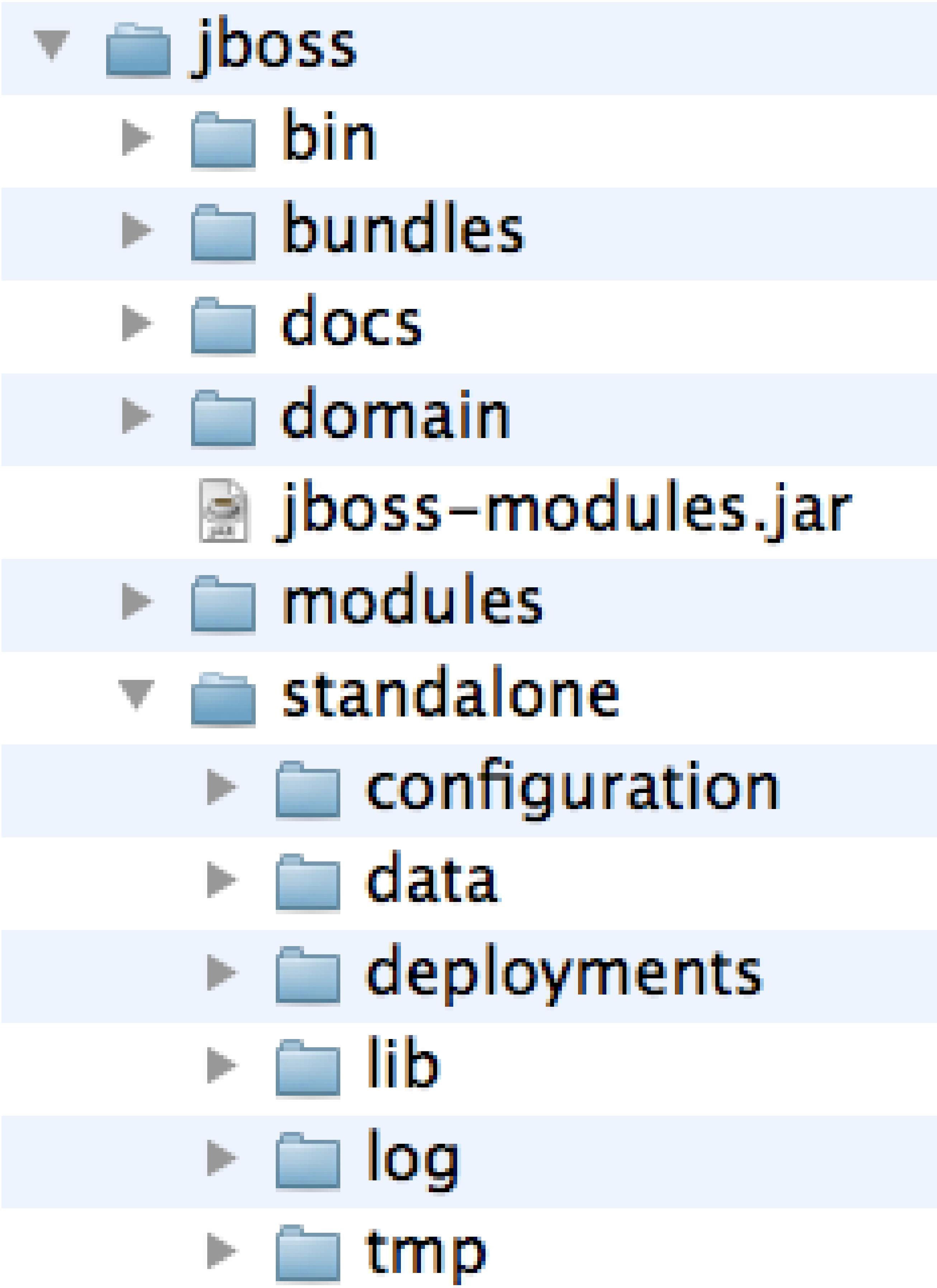
In AS7, all server configuration is kept in two folders for each runtime mode: `standalone` and `domain`. Administrative tasks are simplified from previous releases because all configuration is in one folder and, in standalone mode, in a single file.



TorqueBox uses standalone mode by default but can be run in domain mode as well.

JBoss AS 7 uses a modular architecture - libraries common to all server configurations are kept under the `modules/` directory. Configuration files are stored inside `standalone/configuration/` and `domain/configuration/` folders.

Both standalone and domain modes have a common folder structure, including the following directories: `configuration/`, `deployments/` and `lib/`. In general, it isn't a good idea to remove anything from these directories that you didn't put there yourself.


jboss
bin
bundles
docs
domain
jboss-modules.jar
modules
standalone
configuration
data
deployments
lib
log
tmp

Some additional directories are created automatically at runtime, as needed: `tmp/`, `data/`, and `log/`. Though not typically necessary, you may safely delete these when the server is not running to clear its persistent state.

2. Running

The `$JBOSS_HOME/bin/` directory contains the main JBoss entry point, `standalone.sh` (or `standalone.bat`), along with its config file, `standalone.conf`. Running the JBoss server is simple:

```
$ $JBOSS_HOME/bin/standalone.sh
```

Use the `--server-config` option to specify a different configuration. For example, to put JBoss in "clustered" mode:

```
$ $JBOSS_HOME/bin/standalone.sh --server-config=standalone-ha.xml
```

You may set Java system properties using the `-D` option. Pass `-h` for a list of all the available options.

Permanent runtime configuration of JBoss should go in `bin/standalone.conf`. For example, your application may require more memory (RAM) than the default allocated. Edit `standalone.conf` to increase the value of `-Xmx` to something reasonable.

Though Chapter 19, TorqueBox Capistrano Support doesn't strictly require it, in production you may prefer to control JBoss via a Unix "init script", examples of which may be found in `bin/`. Feel free to tweak one for your particular OS.

We also provide support for managing TorqueBox via upstart [http://upstart.ubuntu.com/]. See Section 4, “Server control” for more details.

3. Deploying

Each runtime mode has a `deployments/` subdirectory, the contents of which determine the applications and services JBoss runs. These apps and services are represented as archives or text files called "deployment descriptors". TorqueBox provides Rake tasks and a `torquebox` to aid the deployment process. For more details, see Chapter 4, TorqueBox Application Deployment and Chapter 5, TorqueBox Deployment Descriptors.

4. Logging

Here we'll focus on the default JBoss configuration and how to expose the JBoss logging system to your Ruby applications, if so desired.

JBoss provides a very sophisticated logging system that nobody completely understands. It's possible to configure hierarchical, categorized log message routing, complex file rotation, syslog integration, SMTP notifications, SNMP traps, JMS, JMX and much more. Obviously, most of that is far beyond the scope of this document.

All JBoss log messages, just like Ruby ones, have an associated level, e.g. DEBUG, INFO, WARN, ERROR, or FATAL. These levels are ordered by increasing severity, e.g. FATAL is higher than ERROR, which is higher than WARN, etc. Unlike messages from a Ruby Logger, each JBoss log message also has a category. Logging configuration rules determine where messages are logged according to their category and level. These rules are contained in the logging subsystem element of `standalone/configuration/standalone.xml`. By default, you will see INFO (and higher, i.e. more severe) messages on the console (the shell where you start TorqueBox) and written persistently to `standalone/log/server.log`.

Anything written to `stdout` or `stderr` is interpreted as an INFO log message and will therefore also be displayed on the console and written to `standalone/log/server.log`.

4.1. TorqueBox::Logger

Of course, standard Ruby Loggers work fine inside of TorqueBox, and you'll find your Rails log files exactly where you expect them to be (unless your app is archived, in which case they'll reside beneath `standalone/log/`). But some users, especially those already familiar with Java logging, may prefer for some Ruby log messages to be passed to JBoss. This is easily achieved using the `TorqueBox::Logger`. For example, you may configure your Rails app like so:

```
config.logger = TorqueBox::Logger.new
```

This results in all Rails-generated log messages being passed to JBoss, hence written to `standalone/log/server.log` in the default configuration. The category for these messages will be the application's name. You can override this by passing the category name in the constructor:

```
TorqueBox::Logger.new( "Billing" )
```

You may pass any type of object to the constructor as long as it responds to `:to_s`. A common convention is to use a class name as a category in Java applications. You can do the same with Ruby:

```
@logger = TorqueBox::Logger.new( self.class )
```

If the class is defined in a module, its name will be something like `YourModule::YourClass`. The logger will replace the double-colons (::) with a period (.), yielding `YourModule.YourClass` as the category. This is because categories are hierarchical and delimited by a period, enabling you to define rules for all classes in the same module, for example.

It's important to understand that, although the `TorqueBox::Logger` does honor the Ruby Logger interface, setting its `:level` attribute will have no effect. The only configuration it honors is JBoss', e.g. the logging subsystem element in `standalone/configuration/standalone.xml`

4.2. JBoss Logging Configuration

The default AS 7.1 configuration is shown below. It includes two handlers (one for the CONSOLE and one for the FILE), some loggers and a root logger. When a message is logged, here's what happens:

1. The message's category is compared to any defined `<logger>` elements. If a match is found, the message's level must be greater than or equal to that of the `<logger>`, else it's discarded.
2. If no matching `<logger>` is found, the message's level must be greater than or equal to the level of the `<root-logger>`, else it's discarded.
3. If it hasn't been discarded, the message is passed to all handlers associated with its `<logger>`, including those in the `<root-logger>`. Set `use-parent-handlers="false"` in the `<logger>` to override this behavior.
4. If the handler has no `<level>` threshold defined, as is the case for the FILE handler below, the message is logged.
5. If the handler has a `<level>` threshold defined, as is the case for the CONSOLE handler below, the message's level must be greater than or equal to that threshold to be logged.

Example 3.1. Logging Config from `standalone/configuration/standalone.xml`

```
          <subsystem xmlns='urn:jboss:domain:logging:1.2'>
              <console-handler name='CONSOLE'>
                  <level name='INFO'/>
                  <formatter>
                           <pattern-formatter pattern='%d{HH:mm:ss,SSS} %-5p [%c]
(%t) %s%E%n'/>
                  </formatter>
              </console-handler>
              <periodic-rotating-file-handler name='FILE'>
                  <formatter>
                           <pattern-formatter pattern='%d{HH:mm:ss,SSS} %-5p [%c]
(%t) %s%E%n'/>
                  </formatter>
                  <file relative-to='jboss.server.log.dir' path='server.log'/>
                  <suffix value='.yyyy-MM-dd'/>
                  <append value='true'/>
              </periodic-rotating-file-handler>
              <logger category='com.arjuna'>
                  <level name='WARN'/>
              </logger>
              <logger category='org.apache.tomcat.util.modeler'>
                  <level name='WARN'/>
```

```
        </logger>
        <logger category='sun.rmi'>
            <level name='WARN'/>
        </logger>
        <logger category='jacorb'>
            <level name='WARN'/>
        </logger>
        <logger category='jacorb.config'>
            <level name='ERROR'/>
        </logger>
        <root-logger>
            <level name='INFO'/>
            <handlers>
                <handler name='CONSOLE'/>
                <handler name='FILE'/>
            </handlers>
        </root-logger>
        <logger category='org.jboss.jca.adapters.jdbc.extensions.mysql'>
            <level name='ERROR'/>
        </logger>
    </subsystem>
```

One thing to note about the default logging configuration is that DEBUG messages won't show up anywhere. So, for example, the following log message will be discarded:

```
@logger = TorqueBox::Logger.new( YourModule::YourClass )
@logger.debug("Will code for food")
```

This is because the "YourModule.YourClass" category matches no logger and its level (DEBUG) is lower than the default level for the root logger (INFO). One solution is to lower the default level for the root logger to DEBUG, but that results in DEBUG messages for every other category that doesn't match any of the loggers, potentially a lot of messages. A better solution is to define a `<logger>` specifically for the category:

```
<logger category='YourModule.YourClass'>
  <level name='DEBUG'/>
</logger>
```

This will result in log messages written to the FILE handler, but not the CONSOLE, since its threshold level is still set at INFO.

For many applications, it's usually better to take advantage of the hierarchical nature of categories and refer only to the module name so that any logger associated with any class within that module will match:

```
<logger category='YourModule'>
  <level name='DEBUG'/>
</logger>
```

For more information, see the official JBoss logging documentation [https://docs.jboss.org/author/display/AS71/Logging+Configuration].

TorqueBox Application Deployment

The TorqueBox Application Server is capable of serving many applications simultaneously. To add your application to the server, you must deploy it. To deploy an application, you tell TorqueBox where to find it. You can do so via our `rake` tasks, the `torquebox` command, or manually.

1. Rake tasks

TorqueBox provides a gem that includes Rake tasks which assist in the deployment to and undeployment from an instance of the TorqueBox Server in addition to other server management tasks. Please see Chapter 18, TorqueBox Rake Support for details.

2. The torquebox command

In addition to the `rake` tasks, TorqueBox provides an executable script that assists in the deployment to and undeployment from an instance of the TorqueBox Server in addition to other server management tasks. Please see Chapter 17, The torquebox Command for details.

3. Manual deployment

To customize some of the aspects of deployment, instead of using the Rake tasks or the torquebox command, you may manually deploy your artifacts. This manual process is identical to the automation that the tasks and command provide.

To deploy manually, you'll need to create two files in the deployment directory, which is by default `$JBOSS_HOME/standalone/deployments/`. The first file is either a small text file (called a deployment descriptor) or a TorqueBox archive, depending on whether you are deploying an application from where it sits on disk or as a self-contained archive. The second file you will need to create is a deployment marker. TorqueBox uses various deployment markers to manage the lifecycle of a deployment.

3.1. Deployment markers

AS7 (and therefore TorqueBox) uses a set of files in the deployment directory called deployment markers to manage the deployment lifecycle of a deployment artifact. A deployment marker for a deployment artifact is an empty file with the same name as the artifact with the marker suffix appended. For manual deployment, you usually only need to be concerned with two of them: `.dodeploy` and `.deployed`. There are quite a few other marker files, and you see the full list in the AS7 docs [https://docs.jboss.org/author/display/AS71/Application+deployment#Applicationdeployment-MarkerFiles].

- `.dodeploy` - signifies that the artifact should be deployed. You would create this file to trigger deployment. TorqueBox removes this marker file after completing the deployment.

* `.deployed` - signifies that the artifact has been deployed. TorqueBox creates this file after completing the deployment.

To trigger redeployment of an already deployed application, simply update the timestamp on its `.deployed` marker file. To undeploy an application, remove the `.deployed` marker file. Example:

```
# redeploy
$ touch $JBOSS_HOME/standalone/deployments/my-app-knob.yml.deployed
# undeploy
$ rm $JBOSS_HOME/standalone/deployments/my-app-knob.yml.deployed
```

3.2. Manually deploying with a descriptor

Applications may be deployed from where they sit on disk. To do so manually, you need to write a deployment descriptor that references the application directory from its `root` entry to `$JBOSS_HOME/standalone/deployments/`. The descriptor is a YAML [http://yaml.org/] file with a required suffix of `-knob.yml`. For details on the various options for authoring deployment descriptors, see Chapter 5, TorqueBox Deployment Descriptors. For TorqueBox to notice your deployment, you'll also need to create an empty `.dodeploy` marker. Example:

```
$ echo "application:\n  root: /path/to/my-app/" > $JBOSS_HOME/standalone/deployments/
my-app-knob.yml
$ touch $JBOSS_HOME/standalone/deployments/my-app-knob.yml.dodeploy
```

3.3. Manually deploying an archive

Ruby web applications may be deployed as atomic self-contained archives. An archive is simply a packaging of the application's directory. The TorqueBox server deploys bundles created with the Java `jar` tool. The Rake tasks and the torquebox command both provide support for the creation and deployment of archives. Please see Section 5, “torquebox archive” or Section 2.2, “Archive-based deployments” for more details.

To manually deploy an archive file, copy it to `$JBOSS_HOME/standalone/deployments/` and create an empty `app-name.knob.dodeploy` marker file in `$JBOSS_HOME/standalone/deployments/`. Example:

```
$ cp something.knob $JBOSS_HOME/standalone/deployments/
$ touch $JBOSS_HOME/standalone/deployments/something.knob.dodeploy
```

3.3.1. Deploying an archive with a deployment descriptor

If you don't want to place your TorqueBox archive in the deployment directory, or need to override some of the configuration options set in the archive, you can instead deploy a deployment descriptor that points to the archive file. Follow the steps described in Section 3.2, “Manually deploying with a descriptor”, but use the path to the archive `.knob` as the `root` instead of the path to the application directory.

4. Zero Downtime Redeployment

TorqueBox now supports redeploying part or all of an application without any downtime. For now, this only works with regular exploded deployments (not archives). To initiate a zero downtime deployment, copy the updated application code to the server (overwriting the currently running files) and touch `$APP_ROOT/tmp/restart.txt`, where `$APP_ROOT` is the root directory of your application. Please be aware that Torquebox will need permissions to read and delete from the `$APP_ROOT/tmp` directory as the restart.txt file will get deleted once Torquebox detects one of the redeployment markers. The application's web runtime will restart, sending all new web requests to the updated runtime while existing requests finish in the old runtimes. Other runtimes may be restarted by touching the appropriate redeployment markers, listed below.

Table 4.1. Zero Downtime Redeployment Markers

Marker File	Runtimes Restarted
`tmp/restart.txt`	web
`tmp/restart-web.txt`	web
`tmp/restart-jobs.txt`	jobs
`tmp/restart-messaging.txt`	messaging
`tmp/restart-services.txt`	services
`tmp/restart-stomplets.txt`	stomplets
`tmp/restart-all.txt`	web, jobs, messaging, services, stomplets

If you don't want to use fileystem markers, runtime pools can also be restarted by JMX operations. Expand the MBean tree under torquebox.pools -> <app_name> to see all the runtime pools for each application. Each pool has a restart operation that behaves identically as touching that pool's restart marker.

TorqueBox Services and Runtime Restarts

Note that when the services runtime gets restarted any running services will be stopped and new service instances initialized and started.

TorqueBox Deployment Descriptors

TorqueBox applications contain one central but optional internal deployment descriptor. A deployment descriptor is simply a configuration file that affects how your components are woven together at deployment time. The internal deployment descriptor used by TorqueBox can be either a YAML text file (known as `torquebox.yml`) or a Ruby file (known as `torquebox.rb`).

The deployment descriptor may be placed inside your application so that it is entirely self-contained. Alternatively, an additional (YAML only) external descriptor may be used outside of your application, overriding portions of the descriptor contained within the application.

Each subsystem within TorqueBox may contribute one or more configurable sections to the descriptor. For more information on the various subsystem descriptor sections, please see: Chapter 6, TorqueBox Web Applications, Chapter 8, TorqueBox Messaging, Chapter 9, STOMP & WebSockets on TorqueBox, Chapter 10, TorqueBox Scheduled Jobs, Chapter 11, TorqueBox Services, Chapter 13, TorqueBox Authentication, and Chapter 16, TorqueBox Runtime Pooling.

1. External and Internal descriptors

Deployment descriptors may be "external", residing outside the application, or "internal", residing within it. The descriptors come in two flavors: YAML-formatted text files and Ruby text files using the TorqueBox configuration DSL. Internal descriptors may use either form, but external descriptors are required to be YAML files.

An external descriptor references an application somewhere on your filesystem. To deploy the application, the descriptor is placed in the `$TORQUEBOX_HOME/jboss/standalone/deployments/` directory of the TorqueBox server. The external descriptor's name should have a suffix of `-knob.yml`.

An internal descriptor should be named `torquebox.yml` or `torquebox.rb` and reside inside the application's `config/` directory, if present, otherwise at the root. Internal descriptors allow you to override the TorqueBox defaults but only for a single app. As such, they are not required. Values in the external descriptor override those in the internal descriptor which, in turn, override the TorqueBox defaults. The YAML syntax used in the external descriptor is identical to the syntax available to the internal descriptor.

2. Contents of a descriptor

There are two syntaxes available for use in an internal descriptor: YAML and a Ruby DSL. For the configuration specifics for each subystem, see: Chapter 6, TorqueBox Web Applications, Chapter 8, TorqueBox Messaging, Chapter 9, STOMP & WebSockets on TorqueBox, Chapter 10, TorqueBox Scheduled Jobs, Chapter 11, TorqueBox Services, Chapter 13, TorqueBox Authentication, and Chapter 16, TorqueBox Runtime Pooling.

2.1. YAML syntax layout

The YAML descriptor has several sections, grouped by subsystem, represented as top-level keys in a YAML associative array.

2.2. Ruby DSL syntax layout

The DSL does not follow the strict sections of the YAML syntax, but the corresponding DSL methods can be grouped and described in the same manner.

To use the DSL, you nest your configuration inside the block passed to `TorqueBox.configure` inside `torquebox.rb`:

```
TorqueBox.configure do
  # DSL calls go here
end
```

The `TorqueBox.configure` method and the DSL methods that take a block can be given a block with or without an argument. If given a block argument, the block will receive a proxy object that must be used to call the DSL methods. Without an argument, the block will use `instance_eval` to evaluate the DSL calls, which will cause issues if you refer to any variables that aren't defined within the scope of the block. Most of the documentation examples use the `instance_eval` (no argument) block syntax.

Using no-argument blocks:

```
TorqueBox.configure do
  web do
    context "/"
  end
end
```

Using argument blocks:

```
TorqueBox.configure do |cfg|
  cfg.web do |web|
    web.context "/"
  end
end
```

You can also mix & match:

```
TorqueBox.configure do
  web do |web|
    web.context "/"
  end

  ruby do
    version "1.9"
  end
end
```

Some DSL methods can also take their options as a hash instead of method calls nested in a block:

```
TorqueBox.configure do
  web :context => "/", :host => "example.com"

  # is equivalent to:

  web do
    context "/"
    host "example.com"
  end
end
```

2.3. General Application Configuration

Location. The application section describes the location for the app itself. Under traditional (mongrel, lighttpd) deployments, this information is picked up through the current working directory. Since the TorqueBox Server runs from a different location, the current working directory has no meaning.

Table 5.1. application

YAML Key	DSL Method	Description	Default
`root`	none	Indicates the location of your application. It may refer to either an "exploded" application (a directory) or the path to a zipped archive. It is required for external descriptors and ignored in an internal descriptor. Rails apps	none

YAML Key	DSL Method	Description	Default
		will have this value set as ENV['RAILS_ROOT'], and Rack apps will have this value set to both the RACK_ROOT constant and ENV['RACK_ROOT'].	

For example, in YAML:

```
application:
  root: /path/to/myapp
```

2.4. Ruby runtime configuration

TorqueBox exposes several of the JRuby runtime options: the ruby compatibility version, the JIT compile mode, and the debug setting. There's also a setting to enable interactive tty for the JRuby runtime for use with the Ruby debugger. All of these options are configured in the ruby: section of a deployment descriptor.

Note that these settings are per application, allowing you to run 1.8, 1.9, and 2.0 applications in the same TorqueBox, or have one JIT'ed and another not.

In a YAML configuration, the ruby settings are grouped under the `ruby` key. For the DSL, they are grouped within the `ruby` block.

Table 5.2. ruby

YAML Key/DSL Method	Description	Default
`version`	The ruby compatibility version for JRuby. Options are: • 1.8 - provides 1.8.7 compatibility • 1.9 - provides 1.9.3 compatibility • 2.0 - provides 2.0.0 compatibility	1.9
`compile_mode`	The JIT compile mode for JRuby. Options are: • jit - Tells JRuby to use JIT on code where it determines there will be a speed improvement	`jit`

YAML Key/DSL Method	Description	Default
	• force - Tells JRuby to use JIT on all code • off - Turns off JIT completely	
`debug`	A value of true enables JRuby's debug logging.	`false`
`interactive`	A value of true sets up the stdin/stdout/stderr of the JRuby runtime for interactive use instead of being redirected to the logging subsystem. Enable this when using the Ruby debugger or the pry gem.	`false`
`profile_api`	A value of true enables JRuby's profiler instrumentation, which allows you to obtain performance information on a given block of ruby code. For more information, check out How to Use the New JRuby Profiler [http://danlucraft.com/blog/2011/03/built-in-profiler-in-jruby-1.6/]	`false`

For example, in YAML:

```
ruby:
  version: 2.0
  compile_mode: off
  debug: false
  interactive: true
  profile_api: true
```

And via the DSL:

```
TorqueBox.configure do
  ruby do
    version "2.0"
    compile_mode "off"
    debug false
    interactive true
```

```
    profile_api true
  end
end
```

2.5. Environment variables

Each application may have its own unique set of environment variables, no matter how many different apps are deployed under a single TorqueBox instance. Variables from internal and external descriptors are merged, with the external variables overriding any internal matching keys.

In a YAML configuration, the environment settings are grouped under the `environment` key. For the DSL, they are grouped within the `environment` block.

For example, in YAML:

```
environment:
  MAIL_HOST: mail.yourhost.com
  REPLY_TO: you@yourhost.com
```

And via the DSL:

```
TorqueBox.configure do
  environment do
    MAIL_HOST 'mail.yourhost.com'
    REPLY_TO 'you@yourhost.com'
  end
end
```

Any variable set in the environment section is accessible from within the application using the ENV hash, e.g. ENV['MAIL_HOST']=='mail.yourhost.com'

2.5.1. Application environment

To set the environment for the application, set either RACK_ENV or RAILS_ENV as an environment variable. They are equivalent, and will both work for a Rack or Rails application. If the application environment is not set, it will default to 'development'. Rails apps will have this value set as ENV['RAILS_ENV'] (which Rails itself will assign to Rails.env), and Rack apps will have this value set to both the RACK_ENV constant and ENV['RACK_ENV'].

For example, in YAML:

```
environment:
  RAILS_ENV: production
```

And via the DSL:

```
TorqueBox.configure do
  environment do
    RAILS_ENV 'production'
  end
end
```

3. Java Deployment Descriptors

3.1. WEB-INF/web.xml

A Java `web.xml` deployment descriptor may be included in your application's `WEB-INF/` directory. Additional Java Servlets, Filters or other configuration may be performed within this file. Its contents will be mixed with other information when your application is deployed. If desired, your `web.xml` may reference the components that TorqueBox implicitly adds.

Rack Filter. TorqueBox provides a Java Servlet™ `Filter` named `torquebox.rack`. This filter is responsible for delegating requests to Rack-based applications.

Static Resource Servlet. In order to serve files from the `public/` directory of your application, TorqueBox installs a `Servlet` named `torquebox.static`.

TorqueBox Web Applications

TorqueBox supports any and all Rack-based web application frameworks, including Ruby On Rails and Sinatra, among others. TorqueBox aims to be unobtrusive, requiring no unusual packaging of your app (e.g. no war files), and unless it depends on obscure native gems, no modifications whatsoever.

So why deploy your Ruby web app on TorqueBox? Because you get cool enterprisey features that every non-trivial app will need eventually if it's successful at all. Let's go over a few.

1. Performance

TorqueBox runs on JRuby, one of the fastest Ruby interpreters available. Because JRuby runs on the Java Virtual Machine, your app runs on real OS threads, so if your app supports multi-threaded invocations, you will make the most of your hardware resources.

Of course, running on the JVM has a drawback: "native" gems that rely upon machine-specific compiled code do not function with JRuby and TorqueBox. You must replace these gems with pure-Ruby or pure-Java implementations. Some native gems using FFI are usable within TorqueBox. Fortunately, gems that won't run on JRuby are becoming more and more rare.

2. Deployment

Most successful web apps evolve to the point that passively responding to HTTP requests is not enough. Before you know it, you may need background processes, scheduled jobs, messaging, and active daemons all in support of your web app.

With TorqueBox these things are an integral part of your app and as such, they share its life cycle. When your application is deployed under TorqueBox, so are your scheduled jobs, background tasks, services, etc. It's simply a matter of editing a single `torquebox.yml` configuration file within your app. This will make your operations staff very happy!

For more details, please see Chapter 4, TorqueBox Application Deployment.

3. Clustering

Clustering nodes is trivially easy (replace 1.2.3.4 with a real IP address):

```
$ $JBOSS_HOME/bin/standalone.sh --server-config=standalone-ha.xml -b 1.2.3.4
```

Or, if using the torquebox-server gem:

```
$ torquebox run --clustered -b 1.2.3.4
```

And when those nodes are behind the JBoss mod_cluster Apache module [http://www.jboss.org/mod_cluster], you get automatic, dynamic configuration of workers, server-side load factor calculation, and fine-grained application lifecycle control.

But even without mod_cluster, TorqueBox clustering provides automatic web session replication and distributed caching, not to mention automatic load-balancing of message delivery, enabling smart distribution of any background processes spawned by your web app.

For more details, please see Chapter 21, TorqueBox Production Tips.

4. Configuration

Ruby web apps are often deployed individually, without respect to hostnames or context-path. Running under TorqueBox, however, you may host several apps under a single host, or multiple apps under different hostnames.

In a YAML configuration, the web settings grouped under the `web` key. For the DSL, they are grouped within the `web` block.

Table 6.1. web

YAML Key/DSL Method	Description	Default
`rackup`	The "rackup" script containing the complete logic for initializing your application.	`config.ru`
`host`	Virtual hosts allow one application to respond to www.host-one.com, while another running within the same JBoss AS to respond to www.host-two.com. This value can be either a single hostname or a YAML list of hostnames.	`localhost`
`context`	Applications within a single TorqueBox Server may be separated purely by a context path. For a given host, the context path is the prefix used to access the application, e.g. http://some.host.com/context. Traditional Ruby web apps respond from the top of a site, i.e. the root context. By using a context path, you can mount applications at a location beneath the root.	`/`

YAML Key/DSL Method	Description	Default
`static`	Any static web content provided by your app should reside beneath this directory.	none unless deploying a Rails application, then `public`.
`session_timeout`	Time (defaults to minutes) for idle sessions to timeout. Specified as an integer followed by a units designation • `ms` designates milliseconds • `s` designates seconds • `m` designates minutes (default if no units are specified) • `h` designates hours	`30m`, specifying 30 minutes.

For example, in YAML:

```
web:
  rackup: alternative/path/to/my_config.ru
  context: /app-one
  static: public
  host: www.host-one.com
```

And via the DSL:

```
TorqueBox.configure do
  web do
    rackup "alternative/path/to/my_config.ru"
    context "/app-one"
    static "public"
    host "www.host-one.com"
  end
end
```

5. Sessions

By using the TorqueBox application-server-based session store, your application gets the benefits of clusterable sessions without having to setup and maintain a database. When clustered, session state is automatically replicated throughout an Infinispan [http://infinispan.org] data grid.

TorqueBox Session Store and Sticky Sessions

It is only recommended to use the TorqueBox session store with a load balancer configured to use sticky sessions. If you want to use round-robin or some other non-sticky load-balancing policy, you may may experience delays as servers wait for the user's session to replicate from other nodes in the cluster. Under high session loads, these delays may turn into timeouts.

Additionally, by using the TorqueBox session store, your application can communicate between both the Java and Ruby sides through the HTTP session. Where possible, elemental scalar attributes of the Ruby session are synchronized to similar attributes in the Java session, and vice-versa.

For complex objects, they are retained in a Ruby hash, and serialized as a blob into a single attribute of the Java session.

When copying between the Ruby and Java sessions, attributes will be retained under symbol keys in the ruby session, and string keys in the Java session. The supported scalar types are numerics, strings, booleans and nil.

How you enable the TorqueBox session store varies based on the web framework used. For Rack applications, just add use TorqueBox::Session::ServletStore to your config.ru. For specific examples in Rails and Sinatra applications, see Section 8.4, “Rails Sessions Configuration” and Section 9.1, “Sinatra Sessions Configuration”.

The session timeout can be configured in your TorqueBox deployment descriptor(s). The general format for the session timeout value is numeric with an optional unit of measure, where `ms` = milliseconds, `s` = seconds, `m` = minutes and `h` = hours. If no unit of measure is provided, minutes will be assumed. Ex: `session_timeout: 10 h` will set the session timeout to 10 hours.

Configuring session timeout, in YAML:

```
...

web:
  host: foobar.com
  context: /tacos
  session_timeout: 10 m
```

And via the DSL:

```
TorqueBox.configure do
```

```
  web do
    host "foobar.com"
    context "/tacos"
    session_timeout "10 m"
  end
end
```

6. Caching

TorqueBox provides an implementation of the Rails 3.x `ActiveSupport::Cache::Store` [http://guides.rubyonrails.org/caching_with_rails.html] that exposes your application to the Infinispan [http://infinispan.org] data grid. Additionally, TorqueBox provides similar functionality for Sinatra sessions. See specific configuration options in the Ruby Web Frameworks sections below. To learn more about the Infinispan cache, and the many other ways it is used by TorqueBox and can be used by you, please see Chapter 7, TorqueBox Caching.

7. Rack

7.1. Rack Applications

Rack is a specification which describes how web server engines can integrate with additional logic written in Ruby. Rack is a akin to CGI or the Java Servlets Spec in terms of goals and functionality.

TorqueBox currently supports general `config.ru`-based applications. In your application's directory, your Rack application can be booted from a file named `config.ru` that you provide. The Ruby runtime provided to your application is quite rudimentary. If you desire to use RubyGems or other libraries, it is up to you to require the necessary files (for instance, `require 'rubygems'`).

```
app = lambda {|env| [200, { 'Content-Type' => 'text/html' }, 'Hello World'] }
run app
```

The directory containing the `config.ru` is considered the current working directory, and is included in the load path.

7.2. Rack API

TorqueBox aims to provide complete Ruby Rack compatibility. Please refer to the Rack specification at http://rack.rubyforge.org/doc/SPEC.html for more information.

Applications implemented by the user must simply provide an object implementing a single-argument method in the form of `call(env)`.

Table 6.2. Rack environment

Variable	Description
`REQUEST_METHOD`	The HTTP request method, such as “`GET`” or “`POST`”. This cannot ever be an empty string, and so is always required.
`SCRIPT_NAME`	The initial portion of the request URL’s “path” that corresponds to the application object, so that the application knows its virtual “location”. This may be an empty string, if the application corresponds to the “root” of the server.
`PATH_INFO`	The remainder of the request URL’s “path”, designating the virtual “location” of the request’s target within the application. This may be an empty string, if the request URL targets the application root and does not have a trailing slash. This value may be percent-encoded when I originating from a URL.
`QUERY_STRING`	The portion of the request URL that follows the `?`, if any.
`SERVER_NAME`	
`SERVER_PORT`	
`HTTP_` variables	Variables corresponding to the client-supplied HTTP request headers (i.e., variables whose names begin with HTTP_). The presence or absence of these variables should correspond with the presence or absence of the appropriate HTTP header in the request.
`rack.version`	The Array [m, n], representing this version of Rack.
`rack.url_scheme`	`http` or `https`, depending on the request URL.
`rack.input`	Input stream
`rack.errors`	Error output stream
`rack.multithread`	Always `true`
`rack.multiprocess`	Always `true`
`rack.run_once`	Always `false`
`rack.session`	
`rack.logger`	Not implemented
`java.servlet_request`	The underlying Java `HTTPServletRequest`

7.3. Sendfile support

With TorqueBox it is possible to delegate the delivery of the response content from a file to the application server itself, without the need to read the file in the application. This is very useful for static files. The `X-Sendfile` header is designed for this usage. You can read more about Rack Sendfile [http://rack.rubyforge.org/doc/Rack/Sendfile.html]. Additionally the TorqueBox implementation allows to use byte-ranges to fetch only part of the file by using the `Range` header.

Sendfile support is available in TorqueBox for all Rack-based applications (including Sinatra and Ruby on Rails frameworks).

It is still possible to use a proxy in front of TorqueBox that handles the `X-Sendfile` header instead of the application server. TorqueBox will not handle the file if a special `X-Sendfile-Type` header is set.

By default the sendfile support is disabled in TorqueBox since it requires native connectors in the JBoss AS web subsystem. To enable the sendfile support in TorqueBox you need to modify the `$JBOSS_HOME/standalone/configuration/standalone.xml` file and change the `native` attribute to `true` in the web subsystem, like this:

```
<subsystem xmlns="urn:jboss:domain:web:1.4" default-virtual-server="default-host"
 native="true">
  ...
</subsystem>
```

This will automatically make the sendfile support available.

7.3.1. Examples

Example 6.1. Application sending content of `thing.txt` file

```
app = lambda { |env|
  [200, { 'Content-Type' => 'text/html', 'X-Sendfile' => 'thing.txt' }, "" ]
}
run app
```

Example 6.2. Receive bytes from 2 to 5 of the file content

```
curl -v -H "Range: bytes=2-5" http://localhost:8080/sendfile/
```

Example 6.3. Receive content of file from beginning up to 10th byte

```
curl -v -H "Range: bytes=-10" http://localhost:8080/sendfile/
```

Example 6.4. Receive content of file from 10th byte to the end

```
curl -v -H "Range: bytes=10-" http://localhost:8080/sendfile/
```

Example 6.5. Multiple ranges

```
curl -v -H "Range: bytes=0-2/10-" http://localhost:8080/sendfile/
```

8. Ruby on Rails

8.1. Ruby on Rails™ Applications

Ruby-on-Rails (also referred to as "RoR" or "Rails") is one of the most popular Model-View-Controller (MVC) frameworks for the Ruby language. It was originally created by David Heinemeier Hansson at 37signals [http://37signals.com/] during the course of building many actual Ruby applications for their consulting business.

Rails has straight-forward components representing models, views, and controllers. The framework as a whole values convention over configuration. It has been described as "opinionated software" in that many decisions have been taken away from the end-user.

It is exactly the opinionated nature of Rails that allows it to be considered a simple and agile framework for quickly building web-based applications. Additionally, since Ruby is an interpreted language instead of compiled, the assets of an application can be edited quickly, with the results being immediately available. In most cases, the application does not need to be restarted to see changes in models, views or controllers reflected.

8.2. Rails 2.3.x versus 3.x

TorqueBox supports both the 2.3.x and 3.x codelines of Rails. By default, all utilities prefer the latest version of a given gem.

8.3. Preparing your Rails application

While TorqueBox is 100% compatible with Ruby-on-Rails, there are a few steps that must be taken to ensure success. The biggest issues to contend with involve database access and native gems. The distribution includes a Rails application template to make the creation or adaptation of a codebase to TorqueBox easier.

8.3.1. Install Rails

Previous releases of TorqueBox bundled Rails but it is no longer included. You'll need to install the version needed by your application.

```
$ gem install rails --version=[rails version]
```

8.3.2. Using the torquebox command line

TorqueBox ships with a command line application which you can use to setup a new Rails application or modify an existing one to work with TorqueBox. The same command works for both scenarios. If the application directory exists, it will apply the TorqueBox rails template to that application. If it does not exist, a new Rails application will be created.

```
$ torquebox rails /path/to/myapp
```

If you have multiple versions of Rails installed, you can generate applications for a specific version by using bundler. For example, you have Rails 2.3.14 and 3.2.2 installed. By default, Rails will use version 3.2.2 when creating a new application. To create a new Rails 2.3.14 application using the TorqueBox template, just create a Gemfile that specifies the Rails version.

```
gem 'rails', 2.3.14'
gem 'torquebox', '3.1.2'
```

To generate the Rails app, first run bundler.

```
$ bundle install
```

Then, when you generate the application, use bundler. It will use the Rails version you specified in your Gemfile.

```
$ bundle exec torquebox rails [myapp]
```

8.3.3. Include the JDBC Gems for Database Connectivity

ActiveRecord applications deployed on TorqueBox benefit from using the Java-based JDBC database drivers. These drivers are provided as a handful of gems which you may include into your application through `config/environment.rb` or a `Gemfile`. For more information on database connectivity within the TorqueBox environment, please see Chapter 14, Database Connectivity in TorqueBox.

When using the TorqueBox Rails application template described above, these modifications are made for you.

Rails 2.x. You simply must reference the `activerecord-jdbc-adapter` from your `environment.rb` within the `Rails::Initializer.run` block.

```
Rails::Initializer.run do |config|

  config.gem "activerecord-jdbc-adapter",
             :require=>'jdbc_adapter'

end
```

All databases will require inclusion of the `activerecord-jdbc-adapter`. No other gems need to be required or loaded, since ActiveRecord will perform further discovery on its own.

Rails 3.x. Rails 3 uses bundler to manage the dependencies of your application. To specify the requirement of the `activerecord-jdbc-adapter` with Rails 3, simple add it to your `Gemfile`. Additionally, any specific JDBC driver your application will require should be indicated. Applications created with Rails 3.1 and later should already include the necessary JDBC gems.

```
gem 'activerecord-jdbc-adapter'
gem 'jdbc-sqlite3'
```

8.4. Rails Sessions Configuration

By default, both Rails 2 and Rails 3 use the simple cookie-based session store, which requires no support from the server. TorqueBox can leverage the cluster-compatible sessions provided by the application server to keep session state on the server. The TorqueBox session store requires no specific configuration of a database or other technology. To use the TorqueBox session store, you must adjust `config/initializers/session_store.rb`. The contents vary depending on the version of Rails your application uses.

In both cases, your application should require the `torquebox` gem, which provides the implementation.

When using the TorqueBox Rails application template described above, these modifications are made for you.

Rails 2.x. In `config/initializers/session_store.rb`

```
ActionController::Base.session_store = :torquebox_store
```

Rails 3.x. In `config/initializers/session_store.rb` (adjust for your application's name)

```
MyApp::Application.config.session_store :torquebox_store
```

If you need more control over the session cookie, the full list of supported options is below.

```
MyApp::Application.config.session_store :torquebox_store, {
  :key => 'my_session_key',
  :domain => 'foobar.com',
  :path => '/baz',
  :httponly => true,
  :secure => true,
  :max_age => 60, # seconds
  :timeout => 180 # seconds
}
```

8.5. Rails Caching Configuration

You configure the TorqueBox cache store the same way you would any other Rails cache store, but we recommend setting it in `config/application.rb` because it will adapt to whichever environment it finds itself. Regardless of its configuration, it will always fallback to local mode when run in a non-clustered, even non-TorqueBox, environment.

In whatever context you use the cache store, you must include the `torquebox` gem, which provides the implementation.

```
module YourApp
  class Application < Rails::Application

    config.cache_store = :torquebox_store

  end
end
```

Using this symbolized form causes Rails to load the appropriate Ruby file for you. Alternatively, you may load the file yourself and then refer to the fully-qualified class name, `ActiveSupport::Cache::TorqueBoxStore`.

By default, the `TorqueBoxStore` will be in asynchronous invalidation mode when clustered and local mode when not. But you can certainly override the defaults:

```
config.cache_store = :torquebox_store, {:mode => :distributed, :sync => true}
```

You can even create multiple cache stores in your app, each potentially in a different clustering mode. You should use the `:name` option to identify any additional caches you create, e.g.

```
COUNTERS = ActiveSupport::Cache::TorqueBoxStore.new(:name => 'counters',
                                                    :mode => :replicated,
                                                    :sync => true)
```

See Chapter 7, TorqueBox Caching for additional details.

8.6. Logging

By default, Rails logs where you would expect, but it's possible to tap into the JBoss log system for more sophisticated logging. For more information, see Section 4.1, “TorqueBox::Logger”.

9. Sinatra

Sinatra [http://www.sinatrarb.com/] is a very simple DSL for creating web applications. And all the TorqueBox features available to Rails apps, e.g. clustering, session replication, and caching, will work for Sinatra app just as well.

9.1. Sinatra Sessions Configuration

Because the TorqueBox session store is Rack compliant, you configure it the same way you would any other session store in Sinatra.

```
require 'sinatra'
require 'torquebox'

class SinatraSessions < Sinatra::Base

  use TorqueBox::Session::ServletStore
```

```
  get '/foo' do
    session[:message] = 'Hello World!'
    redirect '/bar'
  end

  get '/bar' do
    session[:message]   # => 'Hello World!'
  end

end
```

If you need more control over the session cookie, the full list of supported options is below.

```
use TorqueBox::Session::ServletStore, {
  :key => 'sinatra_sessions',
  :domain => 'foobar.com',
  :path => '/baz',
  :httponly => true,
  :secure => true,
  :max_age => 60, # seconds
  :timeout => 180 # seconds
}
```

9.2. Sinatra Caching Configuration

Because the TorqueBox cache store is derived from `ActiveSupport::Cache::Store`, you must include `activesupport-3.x` in your Sinatra app.

In whatever context you use the cache store, you must include the torquebox RubyGem, which provides the implementation.

```
require 'active_support/cache/torque_box_store'
class SinatraCache < Sinatra::Base
  set :cache, ActiveSupport::Cache::TorqueBoxStore.new
end
```

By default, the `TorqueBoxStore` will be in asynchronous invalidation mode when clustered and local mode when not. But you can certainly override the defaults:

```
set :cache, ActiveSupport::Cache::TorqueBoxStore.new(:mode => :distributed, :sync
 => true)
```

You can even create multiple cache stores in your app, each potentially in a different clustering mode. You should use the `:name` option to identify any additional caches you create, e.g.

```
COUNTERS = ActiveSupport::Cache::TorqueBoxStore.new(:name => 'counters',
                                                    :mode => :replicated,
                                                    :sync => true)
```

9.3. Logging

By default, Sinatra log support is minimal, sending most errors to stdout or stderr. For more sophisticated logging, see Section 4.1, “TorqueBox::Logger”.

TorqueBox Caching

1. Overview

As a part of the JBoss AS, TorqueBox utilizes the Infinispan [http://infinispan.org] data grid. Infinispan offers a noSQL key/value store that can be replicated or distributed across a cluster, or run on a single machine locally. The cache is exposed as Ruby modules and classes through TorqueBox. There are two ways that applications can take advantage of this data store.

- `TorqueBox::Infinispan::Cache` Direct Ruby access to the Infinispan cache
- `ActiveSupport::Cache::TorqueBoxStore` Sinatra and Rails session, fragment and other framework caching

Each of these components allows applications to configure the clustering mode and other options of the underlying Infinispan cache, and fully supports JTA and XA distributed transactions in the container.

2. Clustering Modes

Infinispan offers a number of clustering modes that determine what happens when an entry is written to the cache.

Local. This is the default mode when TorqueBox runs non-clustered, roughly equivalent to the Rails `MemoryStore` implementation, though it has some advantages over a simple memory store, e.g. write-through/write-behind persistence, JTA/XA support, MVCC-based concurency, and JMX manageability.

Invalidation. No data is actually shared among the nodes in this mode. Instead, notifications are sent to all nodes when data changes, causing them to evict their stale copies of the updated entry. This mode works best when the infinispan data store is backed by a single, canonical data source such as a relational database. It also works very well for Rails' fragment and action caching, and is the default mode for the cache underlying `ActiveSupport::Cache::TorqueBoxStore`.

Replicated. In this mode, entries added to any cache instance will be copied to all other cache instances in the cluster, and can then be retrieved locally from any instance. This mode is probably impractical for clusters of any significant size. Infinispan recommends 10 as a reasonable upper bound on the number of replicated nodes.

Distributed. This is the default mode for a la carte caches when TorqueBox runs clustered. This mode enables Infinispan clusters to achieve "linear scalability". Cache entries are copied to a fixed number of cluster nodes (2, by default) regardless of the cluster size. Distribution uses a consistent hashing algorithm to determine which nodes will store a given entry.

3. TorqueBox::Infinispan::Cache Options and Usage

The `TorqueBox::Infinispan::Cache` supports a number of options. All components that use the cache as their underlying storage, e.g. `ActiveSupport::Cache::TorqueBoxStore` as well as the `Cache` class itself accept a hash of options. The common options for all cache components are:

Table 7.1. Cache options

Option	Default	Description
`:mode`	`:distributed`	Any of the following will result in replicated mode: • `:r` • `:repl` • `:replicated` • `:replication` Any of the following will result in distributed mode: • `:d` • `:dist` • `:distributed` • `:distribution` Any of the following will result in invalidation mode: • `:i` • `:inv` • `:invalidated` • `:invalidation` Any other value for `:mode` will result in distributed when clustered and local otherwise.
`:sync`	true	The coordination between nodes in a cluster can happen either synchronously (slower writes) or asynchronously (faster writes).
`:name`	{the application's name}	The `:name` option enables you to create multiple cache stores in your

Option	Default	Description
		app, each with different options. It's also a way you can configure multiple apps to share the same cache store.
`:persist`	false	The `:persist` option enables file-based persistence of the cache entries. Any value for `:persist` which is a path to a writable directory will be used for cache storage.
`:transaction_mode`	`:transactional`	By default, all local caches are transactional. If you don't need transactions, set this to `:non_transactional`.
`:locking_mode`	`:optimistic`	Starting with Infinispan 5.1 the supported transaction models are `:optimistic` and `:pessimistic`. The `:optimistic` option defers lock acquisition to transaction prepare time, reducing lock acquisition duration and increasing throughput. With the `pessimistic` model, cluster wide-locks are acquired on each write and released after the transaction completes.
`:encoding`	`:marshal_smart`	The default value provides the widest support for the type of objects you can cache, i.e. any type of Ruby or Java object. Two other possible values for `:encoding` include `:edn` and `:json`. And while limited in the types of data that can be cached with them (essentially just the standard Ruby data types and collections) they can be used to share data among non-Ruby applications with access to the same Infinispan-backed caches.

TorqueBox::Infinispan::Cache Usage. The `Cache` object may be used to store and retrieve values from Infinispan. You can store just about anything: arbitrary Ruby data types, Java class instances, strings, numbers. The gamut. To use the `Cache` just make a new one providing it with initialization options.

```
        require 'torquebox-cache'
       cache = TorqueBox::Infinispan::Cache.new( :name => 'treasure', :persist=>'/
data/treasure' )
```

Adding, removing and updating items in the cache is as you might expect.

```
# Put some stuff in the cache
cache.put( 'akey', "a string value" )
cache.put( "time", Time.now )
cache.put( user.id, user )

# Get it back again
time = cache.get( "time" )
user = cache.get( params[:id] )

# Remove something
cache.remove( 'akey' )
```

You also have typical hash-like methods which allow you to manipulate and query the cache for key information.

```
# Get all of the keys
keyset = cache.keys

# See if the cache contains a key
cache.contains_key? user.id

# Get everything! Caution, this could be very expensive
thewholeshebang = cache.all

# Clear it out. This happens asynchronously, so returns quickly
cache.clear

# Only put this in the cache if there isn't already something there with the same key
```

```
cache.put_if_absent( key, session[:user] )

# And you can replace a value in the cache conditionally
# - if it hasn't changed since the last time you accessed it
t1 = Time.now
t2 = Time.now + 10
cache.put( "time", t1 )
# replaces t1 with t2
cache.replace( "time", t1, t2 )
# does NOT replace since the value is now t2
cache.replace( "time", t1, Time.now + 20 )
```

Increment, Decrement and Transactions. `TorqueBox::Infinispan::Cache` also provides some convenience methods for atomically incrementing or decrementing a value in the cache. Additionally, the `Cache` provides transactional blocks with `Cache#transaction do ...`, and all `Cache` operations automatically participate in XA transactions if they are called from within a `TorqueBox.transaction` block.

```
cache.increment('mykey') # 1
cache.increment('mykey') # 2
cache.decrement('mykey') # 1

# Automatically participates in XA transactions
Torquebox.transaction do
  cache.increment('mykey')
  # ...
end

# And can scope transactions itself
cache.transaction do
  cache.decrement('mykey')
  end
```

To read more about transactions see Chapter 15, TorqueBox Distributed Transactions.

4. ActiveSupport::Cache::TorqueBoxStore Options and Usage

As noted in Chapter 6, TorqueBox Web Applications the TorqueBox store can be used for all of the implicit caching within Rails and session storage within Sinatra. NB:The `TorqueBoxStore` uses

`:invalidation` as it's default mode. If you will be using the Rails cache to explicitly store and retreive values across a cluster you should change the default clustering mode to `:replicated` or `:distributed` mode.

```
config.cache_store = :torquebox_store, {:mode => :distributed}
```

In addition to the common options for `TorqueBox::Infinispan::Cache` as noted above, `ActiveSupport::Cache::TorqueBoxStore` supports all the options of the existing Rails implementations, including the advanced features of `MemCacheStore`, along with a few more to control how data replication occurs amongst the nodes in a cluster.

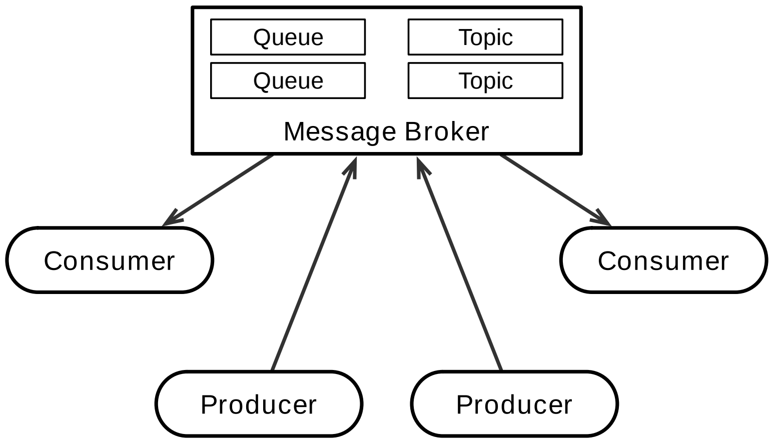
Rails and Sinatra configuration details can be found in Chapter 6, TorqueBox Web Applications. Usage is essentially transparent to the application beyond this configuration.

TorqueBox Messaging

1. Introduction

HornetQ. TorqueBox integrates the JBoss HornetQ message broker technology. It is automatically available to you, with no additional configuration required to start the messaging service. HornetQ supports clustered messaging, to allow for load-balancing, failover, and other advanced deployments.

The term "messaging" encompasses a large area of functionality. Messaging solutions are used to achieve loosely-coupled, asynchronous systems. The primary actors in a messaging-based system are messages, destinations, consumers, and producers. From an implementation perspective, a broker mediates the relationships between the other actors.

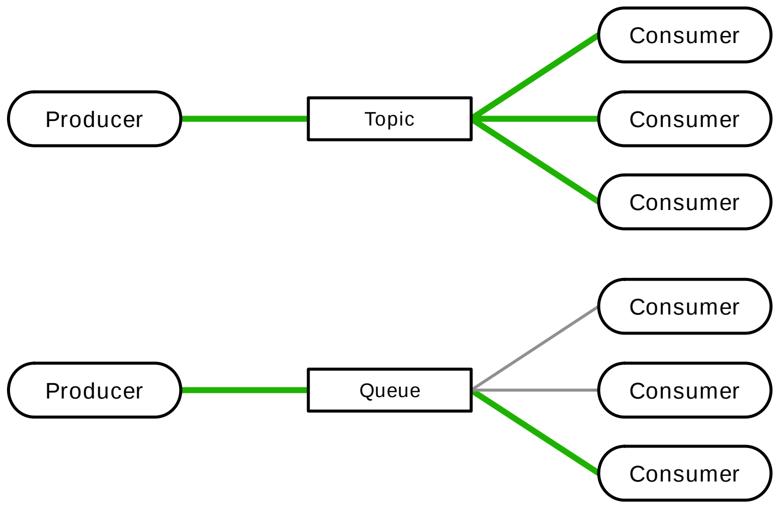


Messages. The unit of communication within a messaging system is a message. A message may either be simply a blob of octets, or it might have some higher-order, application-defined semantics. All messages include a set of headers, similar to email.

Destinations. A destination represents a rendezvous where messages are exchanged. A message may be sent to a destination by one actor, and received from the destination by another.

There are two main types of destinations: queues and topics. All destinations allow multiple actors to place messages with them. The type of destination affects what happens to the message once given to the destination. A queue delivers the message to a single recipient (possibly one of many candidate recipients). A topic delivers the message to multiple interested recipients.

In the image below, the green lines represent the flow of a single message from a producer to one-or-more consumers through a topic and a queue.



Producers. Any component or client code that creates messages and gives them to the message broker for delivery is considered a producer. Generally speaking, the producer does not know the details of the destination.

Consumers. Any component that waits for messages to be delivered to it by the message broker is consider a consumer. A consumer is unaware of the producer and any other consumers, potentially.

2. Deploying Destinations

Queues and topics (collectively known as destinations) may be deployed with your application, or separate from your application. Various parts of your application may also implicitly deploy and use some destinations.

Each method has advantages and disadvantages involving the expectations of your application and its interaction with resources outside the scope of the application.

2.1. Deployment Styles

2.1.1. Deploying destinations with your application

The recommended style is to deploy your queues and topics with your application. This aligns their lifecycle to the deployment cycle of your application. When the app is deployed, its message destinations will be started, if necessary. And when the app is undeployed, its message destinations will be stopped as well, unless they're currently in use by another application. This is by design: messaging destinations are an excellent means of inter-application coordination, so destinations are only stopped when there are no other applications consuming messages from them.

2.1.2. Deploying destinations apart from your application

If you deploy destinations separate and apart from your application, they become long-lived first-class component citizens in your environment. Applications may be deployed and undeployed, while the destinations continue to function, accepting and processing messages to the best of their ability.

If the consumers to a destination are offline, the destination may persist and store any unhandled messages until a consumer re-attaches.

The downside is that by making destinations first-class top-level components of your environment, you must also manage, deploy and undeploy them separate from any app, creating additional work.

2.1.3. Deploying destinations at runtime

You can also choose to deploy messaging destinations at runtime:

Example 8.1. Deploying queues and topics at runtime

```
TorqueBox::Messaging::Queue.start '/queues/foo'
TorqueBox::Messaging::Topic.start '/topics/bar'
```

2.2. Deployment Descriptors

You have several options when deploying queues and topics, based on the lifecycle that suits your systems best.

2.2.1. Application-linked queues and topics

Destinations deployed with your application are configured in your application's deployment descriptor, either its internal one or its external "knob" file.

Within either of these files, you may use a `queues:` section to define queues and a `topics:` section to define topics.

Example 8.2. Defining topics and queues in a deployment descriptor

Using the YAML syntax:

```
application:
  ..
queues:
  /queues/my_app_queue:

topics:
```

```
  /queues/my_app_topic:
```

And via the DSL:

```
TorqueBox.configure do
  ...
  queue '/queues/my_app_queue'
  topic '/queues/my_app_topic'
end
```

2.2.2. Long-lived queues and topics

If your queues and topics have a lifecycle that extends beyond the deployment of your applications, you may want long-lived destinations, which are first-order components, and may be deployed on their own. In this way, many applications can come and go over time, publishing and consuming from the same queues.

When using long-lived destinations, `*-knob.yml` deployment descriptors are placed directly into the `deployments/` directory of TorqueBox AS. Note that a corresponding `*-knob.yml.dodeploy` file is also required in this `deployments/` directory. Simply touch/create a blank file here as part of your deployment.

Queues. To deploy queues, a simple YAML file is required to name the queues and provide additional configuration parameters. The file should have the suffix of `-knob.yml`, such as `these-queues-knob.yml` or `those-queues-knob.yml`. The only configuration option available on queues is the `durable` option.

Durability is enabled by default, and involves writing each message to disk so as to be able to recover in the event of failure or server restart. Disabling durability on queues may result in better performance, but increases the risk of losing messages.

Example 8.3. queues-knob.yml

```
queues:
  /queues/my_queue:

  /queues/my_other_queue:
    durable: false
```

The name of the queue will be used when registering the queue in the naming-service, and is used to discover the queue for attaching consumers and producers.

By convention, queues are named with the prefix of `/queues`.

Topics. To deploy topics, a simple YAML file is required to name the topics and provide additional configuration parameters. The file should have the suffix of `-knob.yml`, such as `these-topics-knob.yml` or `those-topics-knob.yml`. Currently, no additional configuration parameters are allowed - topic durability is controlled via options on the attached processors (See Section 4.8.3, “Connecting Consumers to Destinations”).

Example 8.4. topics-knob.yml

```
topics:
  /topics/my_topic:

  /topics/my_other_topic:
```

The name of the topic will be used when registering the topic in the naming-service, and is used to discover the topic for attaching consumers and producers.

By convention, topics are named with the prefix of `/topics`.

3. TorqueBox Ruby Classes

All classes in the `TorqueBox::Messaging` module reside in the Ruby gem, `torquebox-messaging`, so to use them in your Rails app, you'll need to configure your app to load the gem.

Rails 2.x. Add this to your `config/environment.rb`:

Example 8.5. To use `TorqueBox::Messaging` in a Rails 2.x app

```
Rails::Initializer.run do |config|
  ...
  config.gem 'torquebox-messaging'
  ...
```

Rails 3.x. Add this to your `Gemfile`:

Example 8.6. To use `TorqueBox::Messaging` in a Rails 3.x app

```
source 'http://rubygems.org'

gem 'rails', '3.0.4'
...
```

```
gem 'torquebox-messaging'
```

And to use them in any other JRuby script, it's even simpler. First, ensure that `rubygems` is loaded, then require the `torquebox-messaging` feature.

Example 8.7. To use `TorqueBox::Messaging` in a shell script

```
#!/usr/bin/env jruby

require 'rubygems'
require 'torquebox-messaging'
```

4. Messaging Abstractions

4.1. Queues and Topics

There are two main messaging destination abstractions: `TorqueBox::Messaging::Queue` and `TorqueBox::Messaging::Topic`. Each has a `publish` and a `receive` method, and each must be constructed with a `name` and an optional hash of options:

Table 8.1. Message destination options

Option	Default	Description
`:host`	localhost	Should be the hostname or ip address of the HornetQ server containing the destinations.
`:port`	5445	The port of the HornetQ server.
`:username`	nil	The username to use when connecting to the HornetQ server.
`:password`	nil	The password to use when connecting to the HornetQ server.
`:client_id`		A string to uniquely indentify the connecting client. Optional unless you are using the `:durable` option with `receive` on a `Topic`.

You can also set these options via the `connect_options` on the destination object.

Though sometimes convenient, these methods are fairly low-level and higher-level abstractions such as Message Processors, and Backgroundable are often better-suited to the task.

4.2. Publishing Messages

It's trivial to publish a message to a JMS `Queue` or `Topic` with TorqueBox. And if all of your message consumers are Ruby clients, the contents of the messages can be any serializable Ruby or Java object. You just need to ensure that the type of content you produce resides in the runtime environments of both the producer and the consumer.

To send a message, you will need access to a `Topic` or `Queue` instance. The preferred method for accessing the destination instance is to use `TorqueBox.fetch(...)` (see Messaging Destinations for more details). If you need to pass options to the instance, or only have access to the destination name at runtime, construct either a `Topic` or a `Queue` instance with its name and options. Once you have a destination instance, simply call its publish method. The API's of both Topics and Queues are identical; they each simply represent a destination for your messages.

By default, messages are encoded using Ruby's Marshal serialization mechanism, allowing you to include Ruby objects in your message. If you need to produce messages that will be consumed by non-Ruby or TorqueBox 1.x clients, you can override the encoding mechanism globally or on a per `publish` basis. See Section 4.7, “Message Encodings” for more information.

Example 8.8. Publish text messages

```
queue = TorqueBox.fetch('/queues/foo')
queue.publish "A text message"

topic = TorqueBox.fetch('/topics/foo')
topic.publish "A text message"
```

Example 8.9. Publish a Ruby Hash

```
queue = TorqueBox.fetch('/queues/foo')
queue.publish {:key => 'value', :list => %w{one two three}}
```

This is enormously convenient, as any serializable object is permitted, but it only makes sense if your queue consumers are also written in Ruby.

Example 8.10. Send message using a remote HornetQ server

```
queue = TorqueBox::Messaging::Queue.new('/queues/foo',
                                        :host => 'hornetq_server.mydomain.com',
```

```
                                                :port => 5445)
queue.publish "Some message"
```

The `publish` method takes an optional second argument containing a hash of options:

Table 8.2. Publish options

Option	Default	Description
`:encoding`	:marshal	Specifies the serialization encoding to use for the message. TorqueBox provides the following built-in encodings: • `:marshal` - The message is encoded/decoded via Marshal, and is transmitted as a binary message. • `:marshal_base64` - The message is encoded/decoded via Marshal, and is transmitted as a base64 encoded text message. This was the encoding scheme used in TorqueBox 1.x. • `:json` - The message is encoded/decoded via JSON, and is transmitted as a text message. This encoding is limited, and should only be used for simple messages. • `:edn` - The message is encoded/decoded via the `edn` gem, and is transmitted as a text message. This encoding is most convenient for messages that can be represented using standard Clojure data structures. • `:text` - The message isn't encoded/decoded at all, and is passed straight through as a text message. The content of the message must be a string. See Section 4.7, “Message Encodings” for more information.

Option	Default	Description
`:priority`	:normal	higher priority messages will be delivered before lower priority messages within the context of a queue. You can specify the priority as an integer in the range 0..9, or as one of the following convenience symbols (with the corresponding integer priorities in parentheses): • `:low` (1) • `:normal` (4) • `:high` (7) • `:critical` (9) Higher priority messages will be processed before lower priority ones for a specific message processor.
`:ttl`	0	The maximum time the message will wait in a destination to be consumed, in milliseconds. If the message isn't consumed within this time it will be delivered to an expiry queue. By default, messages don't have a ttl (and therefore never expire). By default, expired messages end up on the `/queue/ExpiryQueue` queue. If you want to do something special with those messages, you'll need to add a processor for that queue.
`:scheduled`	nil	By default a message will be delivered to the queue or topic immediately. You can delay the delivery by using the `:scheduled`. The specified value must be a Time [http://www.ruby-doc.org/core/Time.html] object containing the time it should be delivered. Please note that scheduled messages will be delivered only to

Option	Default	Description
		the consumers connected at the time of publishing the message.
`:tx`	true	By default, messages published within the scope of a transaction will not be delivered until that transaction commits. Set to `false` to override.
`:persistent`	true	By default, queued messages will survive across AS restarts. If you don't want a message to be persistent, set the persistence to `false` (see Section 2.2.2, "`Long-lived queues and topics`" for controlling message durability globally for a queue).
`:correlation_id`	nil	The string value to set for the JMSCorrelationID [http://download.oracle.com/javaee/1.3/api/javax/jms/Message.html#setJMSCorrelationID%28java.lang.String%29] message header.
`:reply_to`	nil	The `javax.jms.Destination` value to set for the JMSReplyTo [http://download.oracle.com/javaee/1.3/api/javax/jms/Message.html#setJMSReplyTo%28javax.jms.Destination%29] message header.
`:type`	nil	The string value to set for the JMSType [http://download.oracle.com/javaee/1.3/api/javax/jms/Message.html#setJMSType%28java.lang.String%29] message header.
`:properties`	nil	A hash of string key/value pairs to set as message properties. This can be used as application-specific headers and matched against in the `:selector` option of the `receive` method.

Option	Default	Description
`:startup_timeout`	30000	The maximum time to wait for the destination to become ready on initial app startup, in milliseconds. On a very slow machine this may need to be increased from the default.

4.3. Receiving Messages

Receiving messages from a JMS `Queue` or `Topic` is very similar to publishing messages. To consume a message, simply construct either a `Queue` or `Topic` instance with its name, and then call its receive method. The API's of both Topics and Queues are identical.

Example 8.11. Receive messages

```
queue = TorqueBox::Messaging::Queue.new('/queues/foo')
message = queue.receive

topic = TorqueBox::Messaging::Topic.new('/topics/foo')
message = topic.receive
```

The `receive` method takes an optional hash of options, described below. It can also take an optional block, to which the received message will be yielded. If a block is supplied, `receive` returns the value of the block. If the block tosses an exception, the broker will consider the message undelivered and will automatically retry delivery up to some configurable limit [10].

Table 8.3. Receive options

Option	Default	Description
`:decode`	true	When `:decode` is set to true, `receive` returns the same value that was sent via `publish`. If `:decode` is false, the JMS javax.jms.Message [http://download.oracle.com/javaee/1.3/api/javax/jms/Message.html] object will be returned instead. This should be true unless you need to access headers or properties of the JMS message.
`:timeout`	0	The amount of time to wait before giving up, in milliseconds. A value

Option	Default	Description
		of 0 means to wait indefinitely. If `receive` times out it will return a `nil` value.
`:selector`	nil	The JMS selector string used to filter messages received by this consumer. For details see the "Message Selectors" section of the javax.jms.Message [http://download.oracle.com/javaee/1.3/api/javax/jms/Message.html] documentation. A `nil` value means all messages are received.
`:startup_timeout`	30000	The maximum time to wait for the destination to become ready on initial app startup, in milliseconds. On a very slow machine this may need to be increased from the default.
`:durable`	false	Specifies that the connection to a topic should be durable. This causes any messages that arrive on the topic to be queued. If false, messages that arrive on the topic when a `receive` is not waiting will be discarded. If `true`, you must also supply a `:client_id` in the connect options for the Topic. This option is ignored for Queues.
`:subscriber_name`	'subscriber-1'	Specifies the subscriber name to be used when creating a durable topic subscription. This option is ignored for Queues and for non-durable receives on a Topic.

4.3.1. Unsubscribing a Durable Topic

If you create a durable topic subscriber by passing the `:durable` option to the `receive` method, that subscription will remain until the HornetQ Topic is shut down. If you no longer need the subscription, you should unsubscribe it by calling the `unsubscribe` method on the `Topic` object. If you provided a `:subscriber_name` to the `receive` call, you will need to provide that same name as an argument to `unsubscribe`.

4.4. Destination management

TorqueBox offers a simple way to manage queues and topics.

4.4.1. Queue management

There are several methods available in the `TorqueBox::Messaging::Queue` to manage the current instance of the queue.

Example 8.12. Queue instance management

```
queue = TorqueBox::Messaging::Queue.new('/queues/vegetables')
queue.paused? # => true
queue.resume
queue.paused? # => false
queue.count_messages # => 300
queue.count_messages('vegetable = tomatoe') # => 30
queue.remove_messages('vegetable = tomatoe') # => 30
queue.count_messages # => 270
queue.stop
```

Table 8.4. Queue instance management options

Method	Params	Description
stop	None	Stops and destroys the queue.
paused?	None	Returns `true` if queue is pause, `false` otherwise.
pause	None	Pauses the queue. Messages put into a queue will not be delivered even if there are connected consumers.
resume	None	Resumes the queue.
remove_messages	`filter`	Removes messages from the queue. The optional `filter` parameter allows to remove only selected messages by filtering them by the properties with a SQL92-like syntax. If no filter is set, all messages will be removed. This method returns the count of removed messages from the queue.

Method	Params	Description
count_messages	`filter`, `destination`	Counts messages in the queue. The optional `filter` parameter allows to count only selected messages by filtering them by the properties with a SQL92-like syntax. If no filter is set, all messages will be counted.
move_messages	`queue_name`, `filter`, `reject_duplicates`	Moves messages from the queue to another queue. The optional `filter` parameter allows to count only selected messages by filtering them by the properties with a SQL92-like syntax. If no filter is set, all messages will be moved. If `reject_duplicates` parameter is set to true it'll reject all duplicates while moving to the other queue.
move_message	`queue_name`, `id`, `reject_duplicates`	Moves the selected message (you can get the `id` like this: `message.jms_message.jms_message_id`) from the queue to another queue. If `reject_duplicates` parameter is set to true it'll reject all duplicates while moving to the other queue.
expire_messages	`filter`	Expires messages from the queue and moves them to the expire address (by default `jms.queue.ExpiryQueue` queue). You can set custom expiry address by using the `expiry_address=` method. The optional `filter` parameter allows to expire only selected messages by filtering them by the properties with a SQL92-like syntax. If no filter is set, all messages will be moved.
expire_message	`id`	Expires the selected message (you can get the `id` like this: `message.jms_message.jms_message_id`) and moves it to the expire address (by default `jms.queue.ExpiryQueue` queue). You can set custom expiry address

Method	Params	Description
		by using the `expiry_address=` method.
send_messages_to_dead_letter_address	`filter`	Sends messages from the queue to the dead letter address (by default `jms.queue.DLQ` queue). You can set custom dead letter address by using the `dead_letter_address=` method. The optional `filter` parameter allows to send only selected messages by filtering them by the properties with a SQL92-like syntax. If no filter is set, all messages will be moved.
send_message_to_dead_letter_address	`id`	Expires the selected message (you can get the `id` like this: `message.jms_message.jms_message_id`) and moves it to the dead letter address (by default `jms.queue.DLQ` queue). You can set custom dead letter address by using the `dead_letter_address=` method.
expiry_address		Using this method you can get or set the expiry address for the current queue. By default it's set to `jms.queue.ExpiryQueue` queue. Make sure you prefix your destination name with `jms.queue.` or `jms.topic.` depending on the type.
dead_letter_address		Using this method you can get or set the dead letter address for the current queue. By default it's set to `jms.queue.DLQ` queue. Make sure you prefix your destination name with `jms.queue.` or `jms.topic.` depending on the type.

4.4.2. Topic management

The `TorqueBox::Messaging::Topic` objects expose only one method to manage the current instance of the topic.

Example 8.13. Topic instance management

```
topic = TorqueBox::Messaging::Topic.new('/topics/vegetables')
topic.stop
```

Table 8.5. Topic instance management options

Method	Params	Description
stop	None	Stops and destroys the topic.

4.5. Listing and looking up destinations

TorqueBox makes it easy to list and lookup any particular queue or topic deployed in the server. There are two class methods available in the `TorqueBox::Messaging::Queue` and `TorqueBox::Messaging::Topic` classes: `list` and `lookup`.

The `list` method allows to list all queues (or topics) deployed in the server. It returns an array of `TorqueBox::Messaging::Queue` (or `TorqueBox::Messaging::Topic`) objects on which you can easily execute management methods mentioned above. If no destinations are available, empty array is returned.

Example 8.14. Queue and Topic list

```
TorqueBox::Messaging::Queue.list.each do |queue|
    queue.name # => '/queues/vegetables'
    queue.count_messages # => 300
    queue.remove_messages('foo = bar') # => 23
    queue.pause
end

TorqueBox::Messaging::Topic.list.each do |topic|
    topic.name # => '/topics/cars'
end
```

The `lookup` method allows to lookup a queue (or topic) by its name. If a particular queue (or topic) is available it returns a `TorqueBox::Messaging::Queue` (or `TorqueBox::Messaging::Topic`) object. If there is no destination found, `nil` is returned.

Example 8.15. Queue and Topic lookup

```
queue = TorqueBox::Messaging::Queue.lookup('/queues/vegetables')
```

```
if queue
    queue.count_messages # => 25
end

TorqueBox::Messaging::Topic.lookup('/topics/cars') # => [Topic: '/topics/cars']
TorqueBox::Messaging::Topic.lookup('/topics/doesntexist') # => nil
```

4.6. Synchronous Messaging

The `publish` and `receive` methods and our higher-level messaging abstractions are designed for asynchronous communication and are recommended for most uses. However, if you do need to send a message and wait for a response, TorqueBox also provides a synchronous messaging abstraction.

Example 8.16. Synchronous messaging

```
queue = TorqueBox::Messaging::Queue.new('/queues/foo')
Thread.new {
  queue.receive_and_publish(:timeout => 5000) { |message| message.upcase }
}
message = queue.publish_and_receive "ping", :timeout => 5000
# message equals "PING"
```

You send a message with the `publish_and_receive` method which blocks until the `:timeout` elapses or a response is received. This method has a default `:timeout` of 10 seconds since you'll rarely want to wait indefinitely for a response. In a separate thread (likely TorqueBox Services - Chapter 11, TorqueBox Services), you consume messages and publish responses with the `receive_and_publish` method. The return value of the block passed to this method is the message response. The options allowed in both these methods are a union of those from `publish` and `receive`. Synchronous messaging is only available with queues, not topics.

4.7. Message Encodings

TorqueBox provides several different encoding serialization schemes for messaging, and allows you to override the default encoding for all of your messages, or override the encoding used on a per `publish` basis. Creating and registering your own encoding is trivial if you need an encoding scheme that is not provided out of the box.

4.7.1. Built-In Encodings

TorqueBox provides the following built-in encodings:

* `:marshal` - The message in encoded/decoded via Marshal, and is transmitted as a binary message. This is the default encoding.
* `:marshal_base64` - The message in encoded/decoded via Marshal, and is transmitted as a base64 encoded text message. This was the encoding scheme used in TorqueBox 1.x.
* `:json` - The message in encoded/decoded via JSON, and is transmitted as a text message. This encoding is limited, and should only be used for simple messages. This encoding is intended to provide interoperability with other languages. Any application that uses the :json encoding will need to provide the json gem via its Gemfile, or, if you are not using Bundler, the json gem must at least be installed.
* `:edn` - The message in encoded/decoded via the `edn` rubygem, and is transmitted as a text message. This encoding is intended to provide interoperability with message producers and consumers written in Clojure. See the Immutant project [http://immutant.org] for more information.
* `:text` - The message isn't encoded/decoded at all, and is passed straight through as a text message. The content of the message must be a string. This is useful for passing messages you can guarantee will always be strings, or you are doing your own application level encoding/decoding.

You can specify the encoding on a per-publish basis (see Section 4.2, “Publishing Messages”), or set the default encoding globally (see Section 4.7.2, “Overriding The Default Encoding”).

4.7.2. Overriding The Default Encoding

You can override the default encoding (:marshal) in your deployment descriptor. This default will be used for any of your publish calls if no encoding is specified at call time. This change will not affect any messages used by TorqueBox internally (to implement Backgroundable for example).

Example 8.17. Overriding the default message encoding

Using the YAML syntax:

```
application:
  ...
messaging:
  default_message_encoding: json
```

And via the DSL:

```
TorqueBox.configure do
  ...
  options_for :messaging, :default_message_encoding => :json
```

```
end
```

4.7.3. Creating Your Own Message Encoding

To create your own message encoding, you need to create a subclass of `TorqueBox::Messaging::Message` that provides `encode` and `decode` methods, along with `ENCODING` and `JMS_TYPE` constants. Below is a simple annotated example of a custom YAML encoding.

Example 8.18. Annotated custom YAML encoding example

```
require 'yaml'

module MyModule
  class YAMLMessage < TorqueBox::Messaging::Message
    # a unique name for the encoding, stored with a published
    # message so it can be properly decoded
    ENCODING = :yaml

    # can also be :bytes for a binary message
    JMS_TYPE = :text

    def encode(message)
      # @jms_message is the actual javax.jms.TextMessage
      @jms_message.text = YAML::dump(message) unless message.nil?
    end

    def decode
      YAML::load(@jms_message.text) unless @jms_message.text.nil?
    end
  end

  # this will register the class under the key given by its ENCODING
  TorqueBox::Messaging::Message.register_encoding(YAMLMessage)
end
```

Using our new encoding:

```
#you'll need to require your encoding class anywhere you publish/receive
require 'yaml_message'

data = [1, 2, 3]
some_queue.publish(data, :encoding => :yaml)
```

```
puts some_queue.receive # [1, 2, 3]
```

For additional examples, see the message classes defined in the TorqueBox source [https://github.com/torquebox/torquebox/tree/master/gems/messaging/lib/torquebox/messaging].

4.8. Message Processors

Message consumers may be implemented in Ruby and easily attached to destinations. A Ruby consumer may either interact at the lowest JMS-level, or take advantage of higher-level semantics.

4.8.1. Low-level message consumption

For the lowest-level implementation of a Ruby consumer, the class must simply implement `process!(msg)` which receives a `javax.jms.Message` as its parameter. Admittedly, this gets quite a lot of Java in your Ruby, but it's available if needed.

Example 8.19. Low-level message consumer

```
class MyLowConsumer
  def process!(msg)
    # manipulate the javax.jms.Message here
  end
end
```

If `process!` raises an exception, the message broker considers the message undelivered and will retry delivery up to some configurable limit (default is 10). If all of those attempts fail, the broker stores the message in a Dead Letter Queue (DLQ) that may be interrogated later.

4.8.2. Syntactic sugar for message consumers

Message consumers may extend `TorqueBox::Messaging::MessageProcessor` and implement an `on_message(body)` method which will receive the body of the JMS message.

Example 8.20. MessageProcessor subclass

```
class MyConsumer < TorqueBox::Messaging::MessageProcessor
  def on_message(body)
    # The body will be of whatever type was published by the Producer
    # the entire JMS message is available as a member variable called message()
  end
  def on_error(exception)
    # You may optionally override this to interrogate the exception. If you do,
    # you're responsible for re-raising it to force a retry.
```

```
  end
end
```

There is an accessor for the actual JMS message that is set by TorqueBox prior to invoking `on_message`, so it's there if you need it.

Just like with `process!`, if `on_message` raises an exception, the message broker considers the message undelivered. You may trap the error by overriding `on_error`, at which point you decide whether to re-raise the exception to force a retry. That is the default behavior if you do not override the method.

4.8.3. Connecting Consumers to Destinations

To connect consumers within a TorqueBox-deployed application, you need to add a messaging: section to your `torquebox.yml` (or external *-knob.yml descriptor), or add a `processor` directive to the destination definition if you are using the DSL (in `torquebox.rb`).

If you are using a YAML descriptor, the messaging: section will contain the mappings from your destinations (topics and queues) to your consumers. The section is a YAML hash, the keys of which are your destination names, which should correspond to existing queues and topics. These destinations may be deployed through the same `torquebox.yml` or as long-lived destinations.

If you are using a DSL descriptor, the consumers are not defined in a separate section, but as part of the queue/topic definition. If the destination is a long-lived destination (managed by another application), then you will need to tell TorqueBox not to try to create the destination by setting the `create` to false.

Example 8.21. Messaging handlers in a deployment descriptor

Using the YAML syntax:

```
application:
  ..
queues:
  /queues/my_app_queue:

messaging:
  /queues/my_app_queue:     MyFooHandler
  /topics/long_lived_topic: MyBazHandler
```

And via the DSL:

```
TorqueBox.configure do
```

```
  ...
  queue '/queues/my_app_queue' do
    processor MyFooHandler
  end

  topic '/topics/long_lived_topic' do
    create false
    processor MyBazHandler
  end
end
```

The classes MyFooHandler and MyBazHandler would correspond to files available on the load path: `my_foo_handler.rb` and `my_baz_handler.rb`, respectively. In a Rails app, these files would typically reside beneath `lib/` or `app/models/`.

The above example shows the simplest possible configuration, but it's possible to alter the behavior of your message processor using the following options:

Table 8.6. Message processor options

Option	Default	Description
`concurrency`	1	May be used to throttle the throughput of your processor. Processors are single-threaded, by default, but you can increase this value to match the number of concurrent messages you want to handle at a time per server. Note that this value determines the number of consumers connected to the destination and thus you'll rarely want a concurrency greater than 1 for topics since that means you'll process duplicate messages. For queues you'll often want this value higher than 1 so you can process messages in parallel.
`singleton`	false	By default, message processors run on all nodes in a cluster, enabling automatic load balancing. Setting this to true results in a single, HA processor, ensuring that only one node in a cluster receives all messages from its associated destination and, with concurrency

Option	Default	Description
		set to 1, in the same order they were published.
`selector`		May be used to filter the messages dispatched to your consumer.
`durable`	false	Turns the processor into a durable subscriber. Once a processor durably subscribes to a topic, if it disconnects any messages sent will be saved and delivered once the processor reconnects. If `true`, you must also supply a `client_id` as well. This setting only affects processors attached to topics, and is ignored for queue processors.
`client_id`		A string to uniquely indentify the connecting client. Optional unless you are using the `durable` option (above) on a `Topic`.
`xa`	false	By default, message processors do not initiate a distributed transaction. Setting this to true will automatically enlist the message processor's receipt of the message and any messaging, cache, or database access inside that processor in single transaction.
`config`		Should contain a hash of data which will be passed to your consumer's constructor, `initialize(Hash)`.
`stopped`	`false`	By default, message processors are started immediately with the application. You can change this behavior and set the parameter to `true` then later use the management methods of message processors like `start` or `stop`.

Example 8.22. Messaging configuration in a deployment descriptor with options set

Using the YAML syntax:

```
application:
  ...
messaging:
  /queues/foo:
    MyFooHandler:
      selector: "cost > 30"
      concurrency: 2
      xa: false
      config:
        type: "premium"
        season: "fall"
  /topics/bar:
    MyBarHandler:
      durable: true
      xa: true
      client_id: my-awesome-client
      stopped: true
```

And via the DSL:

```
TorqueBox.configure do
  ...
  queue '/queues/foo' do
    processor MyFooHandler do
      selector "cost > 30"
      concurrency 2
      xa false
      config do
        type "premium"
        season "fall"
      end
    end
  end

  topic '/topics/bar' do
   processor MyBarHandler, :durable => true, :client_id => 'my-awesome-client', :xa
 => true, :stopped => true
  end
end
```

The YAML and DSL syntaxes enable the configuration to get fairly sophisticated, allowing you to, for example, map a single destination to multiple processors or re-use configuration options in multiple processors. You may never have a need for this much flexibility, but it's available if you do.

Example 8.23. Advanced messaging configuration in a deployment descriptor

Using the YAML syntax:

```
application:
  ...

messaging:
  /topics/simple: SimpleHandler

  /topics/popular:
    - Handler
        concurrency: 5
    - Observer: &defaults
        selector: "x > 18"
        config:
          x: ex
          y: why
    - Processor

/queues/students:
    VerySimpleAnalyzer:
    YouthMonitor:
      selector: "y < 18"
      config:
        h: ache
        i: eye
    LookAndFeel:
      <<: *defaults
```

Here we have `/topics/simple` mapped to a single processor of type `SimpleHandler` using a YAML string, `/topics/popular` mapped to three processors (`Handler`, `Observer`, `Processor`) using a YAML list, and `/queues/students` mapped to three more processors (`VerySimpleAnalyzer`, `YouthMonitor`, `LookAndFeel`) using a YAML hash where each key in the hash corresponds to the processor type. This example also takes advantage of YAML's ability to merge hash's: the `Observer` and `LookAndFeel` processors are configured identically.

And via the DSL:

```
TorqueBox.configure do
  ...
  topic '/topics/simple' do
    processor SimpleHandler
  end

  common_config = { :selector => "x > 18", :config => { :x => 'ex', :y => 'why' } }

  topic '/topics/popular' do |topic|
    topic.processor Handler, :concurrency => 5
    topic.processor Observer, common_config
    topic.processor Processor
  end

  queue '/queues/students' do |queue|
    queue.processor VerySimpleAnalyzer
    queue.processor YouthMonitor do
      selector "y < 18"
      config do
        h 'ache'
        i 'eye'
      end
    end
    queue.processor LookAndFeel, common_config
  end
end
```

Here we have the same configuration as the YAML example above, but expressed via the DSL. Note that we have to use the block argument form for our destinations that share `common_config`. This is due to the no-argument form using `instance_eval`, which does not allow you to access any variables defined outside of the block.

4.8.4. Listing and looking up message processors

TorqueBox makes it easy to list and lookup any particular message processor (including Backgroudables). There are two class methods available in the `TorqueBox::Messaging::MessageProcessor` class: `list` and `lookup`.

Both methods return instances of `TorqueBox::Messaging::MessageProcessorProxy` class which is a proxy for accessing the particular message processor service configuration. The class exposes many methods, like: `concurrency`, `name`, `start`, `stop`. For more information please refer to RDocs for messaging gem.

The `list` method allows to list all message processors deployed with the application. It returns an array of `TorqueBox::Messaging::MessageProcessorProxy` objects. If no message processors are available, empty array is returned.

Example 8.24. Message processor list

```
TorqueBox::Messaging::MessageProcessor.list.each do |processor|
    # Get the concurrency
    processor.concurrency # => 1
    # Get the destination name (queue or topic name)
    processor.destination_name # => "/queues/foo"
    # Get the implementation class name
    processor.class_name # => "SimpleProcessor"
    # Change the message processor concurrency to 3
    processor.concurrency = 3 # => 3
    # Stop the processor
    processor.stop
    # Start the processor
    processor.start
end
```

The `lookup` method allows to lookup a message processor by providing destination and class name (as `String`) of the implementation. If a particular message processor is available it returns a `TorqueBox::Messaging::MessageProcessorProxy` object. If there is no message processor found, `nil` is returned.

Example 8.25. Message processor lookup

```
processor      =      TorqueBox::Messaging::MessageProcessor.lookup('/queues/foo',
 'SimpleProcessor')

if processor
    # Get the name
    processor.name # => "/queues/foo.SimpleProcessor"
    # Get the concurrency
    processor.concurrency # => 1
    # Change the message processor concurrency to 3
    processor.concurrency = 3 # => 3
end
```

4.8.5. Synchronous Message Processors

Message processors by default are asynchronous, but sometimes you would like to receive some value after the message was processed. in TorqueBox you can mark a selected message processor as synchronous which would send the return value of the `on_message(body)` method back to the sender.

Example 8.26. Synchronous message processor

Consider the following simple message processor:

```
require 'torquebox-messaging'

class SynchronousProcessor < TorqueBox::Messaging::MessageProcessor
    def on_message(body)
        "Got #{body} but I want bacon!"
    end
end
```

In this case every received message (we assume a text message in this example) will be transformed and returned back.

You can use the `publish_and_receive` method available in the `TorqueBox::Messaging::Queue` objects to send messages to synchronous message processors and receive the return values.

Example 8.27. Message sender

```
queue = TorqueBox::Messaging::Queue.new('/queues/samplequeue')
queue.publish_and_receive("something") # => "Got something but I want bacon!"
```

As you can see, the message sender receives the return value from the processor.

To make a message processor work synchronously, you need to set the `synchronous` parameter.

Example 8.28. Deployment descriptor configuration

Using the DSL:

```
TorqueBox.configure do
    queue "/queues/samplequeue" do
        processor SynchronousProcessor do
            synchronous true
        end
    end
```

```
end
```

Using YAML:

```
messaging:
    /queues/samplequeue:
        SynchronousProcessor:
            synchronous: true
```

4.9. Remote Message Processors

It is possible to attach a message processor to a remote destination. Such message processors will behave just as a regular message processors with the difference that the queue (or topic) is deployed on a remote host. No changes are required to the message processors itself, but we need to configure the destination and specify where it is located and how to connect to it. Attaching message processors to remote destinations is done in the same way as with local destinations.

Example 8.29. Remote destination configuration

Using the DSL:

```
TorqueBox.configure do
  queue "/queue/remotequeue" do
    remote do
      host "somehost:4444"
      username "username"
      password "password"
    end

    processor SimpleProcessor do
      concurrency 2
    end
  end
end
```

Using YAML:

```
queues:
  /queue/remotequeue:
    remote:
```

```
      host: "somehost:4444"
      username: "username"
      password: "password

messaging:
  /queue/remotequeue:
    SimpleProcessor:
      concurrency: 2
```

Accessing JNDI on remote host

Please note that by default destinations deployed on a host are not visible remotely. To make them visible, you need to export them in a special `jboss/exported` JNDI naming context. You can read more about it in the JBoss AS JNDI Reference [https://docs.jboss.org/author/display/AS71/JNDI+Reference].

TorqueBox makes it easy to add the destination to the exported context. The only thing you need to do is to set the `exported` parameter for selected destination, like this:

Example 8.30. Exporting destinations

Using the DSL:

```
TorqueBox.configure do
  queue "/queue/exported" do
    exported true
  end
end
```

Using YAML:

```
queue:
  /queue/remotequeue:
    exported: true
```

This simple configuration will make the selected queue visible for both remote and local JNDI lookups.

By default security configuration of JBoss AS does not allow to connect to the host and lookup objects in the JNDI tree remotely. You can make it possible by creating a user and setting a password for it. You can use the convenient `$JBOSS_HOME/bin/add-user.sh` script and add an Application User, like this:

```
$ add-user.sh

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): b

Enter the details of the new user to add.
Realm (ApplicationRealm) :
Username : remoteuser
Password :
Re-enter Password :
What roles do you want this user to belong to? (Please enter a comma separated
 list, or leave blank for none)[  ]: remote
About to add user 'remoteuser' for realm 'ApplicationRealm'
Is this correct yes/no? yes
Added user 'remoteuser' to file '/work/torquebox/jboss/standalone/
configuration/application-users.properties'
Added user 'remoteuser' to file '/work/torquebox/jboss/domain/configuration/
application-users.properties'
Added user 'remoteuser' with roles remote to file '/work/torquebox/jboss/
standalone/configuration/application-roles.properties'
Added user 'remoteuser' with roles remote to file '/work/torquebox/jboss/
domain/configuration/application-roles.properties'
Is this new user going to be used for one AS process to connect to another AS
 process?
e.g. for a slave host controller connecting to the master or for a Remoting
 connection for server to server EJB calls.
yes/no? yes
To represent the user add the following to the server-identities definition
 <secret value="OBNEMSIzIUA=" />
```

Now you can put the credentials into the deployment descriptor as shown above.

4.10. Backgroundable Methods

TorqueBox also provides `Backgroundable` methods. `Backgroundable` allows you to process any method on any class or object asynchronously. You can mark a method to always execute in the background, or send a method to the background on an ad hoc basis. Backgrounded methods return a Future object that can be used to monitor the status of the method invocation and retrieve the final return value. When transitioning from TorqueBox 1.x to 2.x, it is advisable to replace any `Task` implementation with usage of `Backgroundable`.

4.10.1. A note on receiver/argument marshalling

We serialize the receiver and arguments using Marshal and include them in the message that gets enqueued. The message processors run in a separate ruby runtime from the application, which may be on a different machine if you have a cluster. The marshaling works well for basic ruby objects. It may not work as well for objects that expect a lot of plumbing in place (ActionControllers, for example). If you are using instances with a backing store (ActiveRecord instances, for example), it's generally a better idea to send the id of the instance instead of the full instance, and to look it up from the backing store at the start of the backgrounded method's invocation.

4.10.2. `always_background`

`Backgroundable` provides the `always_background` class method that allows you to flag a class or instance method to always be executed in the background:

Example 8.31. Having a method always execute in the background

```
class User < ActiveRecord::Base
  always_background :send_signup_notification

  def self.send_signup_notification(user_id)
    ...
  end
end

# executes in the background, returning immediately
future_result = User.send_signup_notification(42)
```

The `always_background` method can be called before or after the method being backgrounded is defined, and can take multiple method symbols: `always_background :foo, :bar`.

You can also call `always_background` from outside of the class definition if you prefer:

Example 8.32. Alternative `always_background` usage

```
class User < ActiveRecord::Base
  def self.send_signup_notification(user_id)
    ...
  end
end

User.always_background(:send_signup_notification)
```

4.10.2.1. Using always_background with instance methods

`always_background` can be used to enable backgrounding for class or instance methods, even in the same invocation:

Example 8.33.

```
class User < ActiveRecord::Base
  always_background :a_class_method, :an_instance_method

  def self.a_class_method
    ...
  end

  def an_instance_method
    ...
  end
end
```

Even though you can background instance methods, we generally recommend you use class methods for backgrounding where possible, especially if the backing store for an instance can change between when the backgrounded call is queued and when it is executed. See the example below for the preferred method for dealing with such instances (we'll use an `ActiveRecord` class for the example, but the same applies for any instance with a backing store):

Example 8.34.

```
class User < ActiveRecord::Base
  always_background :process_avatar_for

  def process_avatar
    ...
  end

  def self.process_avatar_for(id)
    # pull the user from the db, in case it has changed since this
    # call was backgrounded
    User.find(id).process_avatar
  end
end
```

```
user.avatar = some_file
User.process_avatar_for(user.id)
```

4.10.3. background

If you have not marked a method with `always_background`, you can background it at call time with the `background` method. A method called via `background` will also return a Future object that can be used to monitor the status of the method invocation and retrieve the final return value. `background` can be used with class or isntance methods.

Example 8.35. Backgrounding a method ad hoc

```
class User < ActiveRecord::Base
  def do_something
    ...
  end

  def self.do_something_else
    ...
  end
end

user_instance = User.find(id)

# executes in the background, returning immediately
future_result = user_instance.background.do_something

# executes in the foreground (this thread)
regular_result = user_instance.do_something

# executes in the background, returning immediately
future_result = User.background.do_something_else

# executes in the foreground (this thread)
regular_result = User.do_something_else
```

The same caveats listed above for using `always_background` with instance methods apply to `background` as well.

4.10.4. The Backgroundable module

To use `Backgroundable` methods in a class, you will need to include the `TorqueBox::Messaging::Backgroundable` module into the class:

Example 8.36. Including the `Backgroundable` module

```
class User
  include TorqueBox::Messaging::Backgroundable
  ...
end
```

Including `Backgroundable` provides both the `always_background` class method and `background` as both a class and an instance method.

If your appplication uses Rails and you use the rails template that ships with TorqueBox to create or update your application (`torquebox rails my_app`), you should have an initializer (`RAILS_ROOT/config/initializers/active_record_backgroundable.rb`) that already includes `Backgroundable` into `ActiveRecord::Base`.

4.10.5. `Backgroundable` method invocation options

The priority, time-to-live (TTL), transaction, and persistence options that are available when publishing messages are available to `Backgroundable` methods as well:

Example 8.37. Passing options to `Backgroundable` methods

```
class Widget
  always_background :productize, :priority => :low
  def self.productize
    ...
  end

  def self.monetize
    ...
  end
end

Widget.background(:ttl => 1000, :persistent => false).monetize
```

The message options are passed as a `Hash` as the last argument to `always_background`, and as the only argument to `background`. Options passed to `always_background` affect every background invocation of the specified methods, while options passed to `background` affect only that particular invocation.

Additionally you can define the `future_ttl` option which sets the time (in miliseconds) to wait before moving the not received future to the expiry queue. By default it's set to 600000 (10 minutes).

4.10.6. `Backgroundable` message processor options

All of the options available to message processors in a deployment descriptor are available to `Backgroundable` message processors, too, though possibly only the `concurrency` option is applicable. In addition to the processor options, you can also specify the durability of the underlying queue. This `durable` option is different than the `durable` option for topic processors, and instead controls the durability of the underlying queue. It is `true` by default.

Example 8.38. Backgroundable message processor options in a deployment descriptor

Using the YAML syntax:

```
application:
  ...
tasks:
  Backgroundable:
    concurrency: 2
    durable: false
```

And via the DSL:

```
TorqueBox.configure do
  ...
  options_for Backgroundable, :concurrency => 2, :durable => false
end
```

By default, every application you deploy will have a queue for `Backgroundable` methods, even if you don't use it. To turn off the queue, set the `concurrency` to 0.

4.11. Future Objects

Methods backgrounded via `Backgroundable` return `Future` objects that allow you to monitor the progress of the asynchronous processing.

Table 8.7. Future instance methods

Method	Description
started?	Returns `true` if the task processing has started.

Method	Description
complete?	Returns `true` if the task processing has completed without error. If `true`, The result is available via the `result` method.
error?	Returns `true` if an error occurred during the task processing. If `true`, The actual error is available via the `error` method.
status	Returns the last status message returned from the task. This will only have meaning if you signal status information from within your task. See the status notifications section for more details.
status_changed?	Returns true if the status has changed since you last called `status`. This will only have meaning if you signal status information from within your task. See the status notifications section for more details.
all_statuses	Returns an array of all the statuses received by the future, which may not include all of the statuses sent if the task completes before they are all received. This will only have meaning if you signal status information from within your task. See the status notifications section for more details.
result	Returns the result of the remote processing. This method takes a `timeout` (in milliseconds), and will block for that amount of time if processing has started but not completed, or up to twice that time if processing has yet to start. If no result is available after timing out, a `TorqueBox::Messaging::TimeoutException` is raised. The `timeout` defaults to 30 seconds. The recommended pattern is to wait for `complete?` to return `true` before calling `result`.
method_missing	Delegates any missing methods to the `result`, using the default timeout.
error	Returns the remote error object if an error occurred during task processing.

4.11.1. Sending status notifications to the Future from within the task

From within a task or backgrounded method invocation, you can send a status notification to the `Future` for this call by using the `future.status=` method. The status can be any marshalable object, and its semantics are defined by your application.

Please note that the `future.status=` method is the only method available on the `future` object inside the task or backgrounded method since it is a `FutureProxy` instance.

Example 8.39. Sending a status message

```
class Something
  include TorqueBox::Messaging::Backgroundable

  always_background :process_some_stuff
  ...
  def self.process_some_stuff
    stuff.each_with_index do |thing, index|
      thing.process_it
      # report the % complete
      future.status = (index * 100)/stuff_count
    end
  end
end

f = Something.process_some_stuff
# time passes
f.started? # => true
f.status   # => 22
# time passes
f.status   # => 87
```

STOMP & WebSockets on TorqueBox

1. Overview

TorqueBox provides real-time bidirectional communication between applications and web-browsers using a combination of WebSockets and STOMP. Raw access to WebSockets is not provided. Instead, multiplexed communication is supported through the layering of messaging semantics on top. Additionally, optional integration into other messaging systems (such as JMS/HornetQ) are provided to enable advanced application architectures.

TorqueBox provides support for Stomplets to allow explicit control and design of messaging end-points, instead of simple direct bridging to some other underlying messaging technology, such as a JMS broker.

1.1. What is WebSockets?

WebSockets is a new specification to allow synchronous bidirectional communication between a client (such as a web browser) and a server. While simliar to TCP sockets, WebSockets is a protocol that operates as an upgraded HTTP connection, exchanging variable-length frames between the two parties, instead of a stream.

A browser may access a WebSockets-based service using Javascript. Once connected, the client and server must determine the meaning of any data sent across the socket. The WebSockets transport itself provides no protocol semantics beyond data frames passing each direction. TorqueBox implicitly applies STOMP semantics to the WebSocket connection.

1.2. What is STOMP?

STOMP stands for Stream-Oriented Messaging Protocol. STOMP defines a protocol for clients and servers to communicate with messaging semantics. STOMP does not define any implementation details, but rather addresses an easy-to-implement wire protocol for messaging integrations.

STOMP provides higher semantics on top of the WebSockets transport. STOMP defines a handful of frame types that are mapped onto WebSockets frames.

- `CONNECT`
- `SUBSCRIBE`
- `UNSUBSCRIBE`
- `SEND` (messages sent to the server)
- `MESSAGE` (for messages send from the server)

- `BEGIN`, `COMMIT`, `ROLLBACK` (transaction management)

1.3. What are Stomplets?

The Stomplet specification defines a controller (in the MVC sense of controllers) API for working with asynchronous messaging end-points. Stomplets are mapped to STOMP destinations (possibly using wildcards, like Rails routes), coordinating clients subscribing to receive messages and clients sending messages.

Stomplets are long-lived stateful controllers.

2. Ruby Stomplets

Ruby Stomplets have a handful of methods which may be implemented to support all messaging actions.

- `configure(config)` Configures the Stomplet with its name/value configuration and context.
- `destroy()` Destroys the Stomplet and releases resources when it is taken out of service.
- `on_subscribe(subscriber)` Called when a client wishes to receive messages.
- `on_unsubscribe(subscriber)` Called when a client no longer wishes to receive messages.
- `on_message(message, session)` Called when a client has sent a message.

2.1. Stomplet API

2.1.1. `configure(config)`

The configure(config) method is called for each instance of the Stomplet instantiated by the container. The config parameter includes any name/value pairs specified in the configuration of the Stomplet for a given route.

The configure(...) method is typically where a Stomplet would acquire any resources it needs to handle subscription requests and sent messages.

2.1.2. `destroy()`

The `destroy()` method is called for each instance of the Stomplet when the container undeploys its route. This method is typically where all resources are released and connections to underlying systems are terminated.

2.1.3. `on_subscribe(subscriber)` and `on_unsubscribe(subscriber)`

The `on_subscribe(subscriber)` method is called when a client wishes to receive messages from a destination matching the Stomplet. The same instance of the subcriber parameter is passed to `on_unsubscribe(...)` when the client wishes to cease receiving messages and cancel that subscription.

The subscriber object supports a few useful methods:

- `destination` String describing the desired destination to receive messages from.
- `send(message)` Deliver a message to the client through this subscription.
- `session` Access to the STOMP session (see below).
- hash-like syntax (using `subscriber[paramName]`) to access named parameters from matched routes.

2.1.4. `on_message(message, session)`

The on_message(message) method is called with a Stomp message and session as the parameters whenever a client sends a message to a destination handled by the Stomplet. The client does not necessarily need to have previous subscribed to the destination in order to send messages to it.

2.1.5. Sessions

If the Stomplets are run as part of a larger application which involves web components (Rack, Sinatra, Rails, etc), then the user's session will pass between the web and STOMP components. Values set on the session from a web-based controller will be visible within the scope of the Stomplet, and vice-versa.

If the Stomplets are deployed without a web component, or using different virtual-host configuration, a STOMP-specific session will be used, providing communication between all of the Stomplets but independent from any web sessions.

2.2. Example

```
require 'torquebox-stomp'

class SimpleBroadcastStomplet

  def initialize()
  end

  def configure(stomplet_config)
  end

  def on_message(stomp_message, session)
    @subscribers.each do |subscriber|
```

```
    subscriber.send( stomp_message )
  end
end

def on_subscribe(subscriber)
  @subscribers << subscriber
end

def on_unsubscribe(subscriber)
  @subscribers.delete( subscriber )
end

end
```

3. JMS Integration

TorqueBox provides useful classes upon which you can build your own application's Stomplets. The most useful of these is `TorqueBox::Stomp::JmsStomplet`, which handles a large portion of bridging between STOMP and JMS, while allowing the flexibility to adapt the integration to match your particular needs.

The primary assistance it provides is through two methods:

- subscribe_to(subscriber, jms_destination, jms_selector=nil)
- send_to(jms_destination_name, stomp_message, headers={})

Your own Stomplet may use these methods to handle the heavy-lifting after translating between STOMP destinations and JMS destinations.

When using `send_to(...)`, the `stomp_message` parameter may be a complete `StompMessage`, or simply a string, which will be converted into a message. Any headers specified will override any headers provided through the `StompMessage`.

Example 9.1. Example JMS Stomplet Bridge

```
require 'torquebox-stomp'

class BridgeStomplet < TorqueBox::Stomp::JmsStomplet

  def initialize()
    super
  end

  def configure(stomplet_config)
```

```
    super

    @destination_type = stomplet_config['type']
    @destination_name = stomplet_config['destination']
  end

  def on_message(stomp_message, session)
   send_to( destination_for( @destination_name, @destination_type ), stomp_message )
  end

  def on_subscribe(subscriber)
              subscribe_to(   subscriber,   destination_for(   @destination_name,
 @destination_type ) )
  end

end
```

3.1. Destination and Message compatibility

When using the `JmsStomplet` to bridge STOMP destinations to JMS destination, normal message-encoding occurs. This allows your application to send a message to a JMS destination using `TorqueBox::Messaging` interfaces as normal. The `JmsStomplet` will appropriately decode the messages received from the JMS destination. Likewise, any messages sent by the `JmsStomplet` will be appropriate encoded in order to be consumable by other non-STOMP `MessageProcessors`.

4. Deployment descriptors

To deploy Stomplets with your application, a `stomp` section is added to your application's `torquebox.yml` or `torquebox.rb` descriptor. The section should contain named sections for each Stomplet your application needs to deploy. Each Stomplet is bound to a route, which works similar to Rails request routing, but matches against STOMP destinations instead of web URLs. Additionally, it specifies the class of the implementation, along with optional configuration in the form of name/value pairs of strings.

Parameters from flexible routes may be accessed within the `on_subscribe()` and `on_unsubscribe()` methods using a hash-like syntax: `subscriber[paramName]`.

STOMP supports the notion of virtual hosts, just as with web container. By default, if your application specifies a virtual host for the web portion of the configuration, the same value will be used for the STOMP container. The host may be overridden, though, by specifying a `host:` parameter within the `stomp:` block.

To configure stomplets using the YAML syntax:

```
stomp:
  host: somehost.com
  stomplets:
    stomplet.one:
      route: '/queues/:queue_name'
      class: StompletOne
    foo.stomplet:
      route: '/bridge/foo'
      class: BridgeStomplet
      config:
        type: queue
        destination: /jms-queues/foo
    bar.stomplet:
      route: '/bridge/bar'
      class: BridgeStomplet
      config:
        type: topic
        destination: /jms-topics/bar
```

To configure stomplets via the DSL:

```
TorqueBox.configure do
  ...
  stomp do
    host'somehost.com'
  end

  stomplet StompletOne do
    route '/queues/:queue_name'
  end

  stomplet BridgeStomplet do
     name 'foo.stomplet' # required if >1 stomplets use the same class, optional
 otherwise
    route '/bridge/foo'
    config do
      type 'queue'
      destination '/jms-queues/foo'
    end
  end

  stomplet BridgeStomplet do
```

```
    name 'bar.stomplet' # required if >1 stomplets use the same class, optional
 otherwise
    route '/bridge/bar'
    config :type => 'topic', :destination => '/jms-topics/bar'
  end
end
```

5. Javascript Client

TorqueBox makes use of the Stilts framework and to implement the WebSockets and STOMP stack. TorqueBox includes the Javascript client provided by the Stilts distribution in the `share/javascripts/` directory. The client is derived from work by Jeff Mesnil. The client has several methods of connecting back to the server, transparent to your application:

1. Pure WebSockets
2. WebSockets over Flash (using the Gimit web-socket-js library)
3. HTTP POST for client-to-server with Server-Sent-Events (SSE) providing server-to-client
4. HTTP POST for client-to-server with lingering HTTP GET (long polling, or Comet) providing server to client

5.1. Using the Javascript client

The Javascript STOMP client isolates your application from the underlying WebSocket protocol. The Javascript client works purely in terms of STOMP semantics. The client is based around callbacks.

5.1.1. Instantiating a client

The client is created using the `Client(...)` constructor of the Javascript `Stomp` class. This method takes up to three parameters, being the host, port and secure flag of the server to connect to. Creating the client does not connect it. The client is contained in a file named `stomp.js`, which can be served directly from the Stomplet server port, by requesting `/stomp.js`. When the STOMP client is loaded in this fashion into your HTML page, the Client() constructor is pre-configured with the host, port and secure flag for the same server. See Section 5.2, “Injecting the endpoint URL” for details on retrieving the endpoint variable used in the example below.

```
// Load the client from the STOMP server
<script src="http://<%= endpoint.host %>:<%= endpoint.port %>/stomp.js"></script>

<script type="text/javascript">
  // To use the preconfigured client provided by the server.
  client = new Stomp.Client();
```

```
  // To specify the host, port and secure flag explicitly
  client = new Stomp.Client( "somehost.com", 8676, true );
</script>
```

5.1.2. Connecting the client

To connect the client to the STOMP server, use the connect(...) method, which takes up to three arguments: username, password and a callback function which will be invoked once the connection has been established.

Currently the username and password parameters are ignored and can be skipped, allowing just the specification of the callback function. Typical applications will authenticate the user through traditional web-based logins, storing an authorization token in the session, which is then available to each Stomplet to positively identify the connected client.

```
client.connect( function() {
    // executed once successfully connected
} );
```

Once connected, the callback function will be invoked. Other client methods should be used from within this method or other callbacks.

5.1.3. Sending a message

Messages may be sent to any destination supported by your Stomplets, even without prior subscription to the same destination.

The client's `send(...)` method is used to deliver a payload with headers to the destination.

```
client.send( "/some/destination", { header1: 'Header 1' }, "this is the payload" );
```

Messages sent this way will by processed by the `on_message(...)` method of the Stomplet bound to the destination.

5.1.4. Subscribing to destinations

A client can subscribe to STOMP destinations using the subscribe(...) method, passing the destination and a message-handling callback function as parameters. Any message delivered to the client on the destination will invoke the function with the message as the argument. The message provides access to the body and hash of headers, along with an `ack()` method to acknowledge receipt, if required.

```
client.subscribe( "/some/destination", function(message) {
  // message.body
  // message.headers['header1']
} );
```

5.1.5. Working with transactions

The STOMP protocol defines transactional semantics, and several transactions may concurrently be in use at the same time.

Starting a transaction. The `begin()` method is used to start a transaction. It returns a transaction identifier which may be used to associate other activities with the transaction.

Committing a transaction. To commit the work performed within the scope of a particular transaction, the `commit(...)` method is used, passing the transaction identifier provided by `begin()` as the only parameter.

Aborting a transaction. To cancel the work performed within the scope of a particular transaction, the `abort(...)` method is used, passing the transaction identifier provided by `begin()` as the only parameter.

Sending messages within a transaction. To send a message within a transaction, a `transaction` header should be added, with its value being the transaction identifier returned by a previous call to `begin()`.

```
tx = client.begin();

client.send( "/some/destination", { transaction: tx }, "this is a transactional
 message" );

if ( everyoneIsHappy ) {
  client.commit( tx );
} else {
  client.abort( tx );
}
```

5.2. Injecting the endpoint URL

In your application, it's useful to be able to know exactly the STOMP server's endpoint URL, without having to hard-code that into your application. Since JBoss AS allows changing ports for services, especially when running multiple nodes on the same machine, determining the URL at runtime is helpful. The STOMP endpoint supports many protocols, therefore it is up to the application to construct an appropriate URL involving `stomp://`, `stomp+ssl://`, `ws://`, `wss://`, `http://`, or `https://` URLs. You may use HTTP-centric URLs to load the Javascript client. The Javascript client needs only the host, port and secure flag to connect, figuring which protocols to use as necessary.

You may inject stomp-endpoint into your application code, and it will provide the the details for connections to the STOMP endpoint on that node.

```
endpoint = TorqueBox.fetch('stomp-endpoint')

puts endpoint.host
puts endpoint.port
puts endpoint.secure?
```

Additionally, to fetch the secure endpoint (if provisioned) you may use the key `stomp-endpoint-secure`.

6. Other Clients (without WebSockets)

The Stilts distribution also includes JRuby-based clients and Java clients appropriate for communicating with the TorqueBox STOMP service. While STOMP is offered over WebSockets, the same service, on the same port (8675) provides bare STOMP also, for clients not requiring a WebSockets transport. The JRuby and Java clients can seamlessly communicate with the TorqueBox STOMP server using either TCP/IP, or WebSockets as the underlying transport.

7. Secure connections (TLS and wss://)

If the web container is configured to provide an HTTPS/SSL connector (see SSL Configuration How-To [http://docs.jboss.org/jbossweb/7.0.x/ssl-howto.html]), then a matching secure connector for WebSockets will be provisioned. By default, the secure connector runs on port 8676. WebSocket implementations in browsers do not typically provide a method for accepting untrusted SSL certificates, so it is necessary to first ensure that any self-signed certificate is designated as trusted by the browser before attempting to initiate a secure WebSocket connection. To make a secure connection, clients should use the `wss://` scheme instead of simply `ws://` in their connection URLs.

Additionally, the secure connector can accept pure-STOMP client connections (not using WebSockets), if TLS is enabled on the client.

8. Further information

TorqueBox uses the Stilts project to provide the WebSockets and STOMP stack. The Stilts project also defines the Stomplet API. Additional clients are available directly from the Stilts project. Rest assured, Stilts is written by the same people who write TorqueBox.

http://stilts.projectodd.org/

TorqueBox Scheduled Jobs

1. What Are Scheduled Jobs?

Scheduled jobs are simply components that execute on a possibly-recurring schedule instead of in response to user interaction. Scheduled jobs fire asynchronously, outside of the normal web-browser thread-of-control. Scheduled jobs have full access to the entire Ruby environment. This allows them to interact with database models and other application functionality.

2. Ruby Job Classes

Each scheduled job maps to exactly one Ruby class. The path and filename should match the class name of the job contained in the file.

File name	Class name
`mail_notifier.rb`	MailNotifier
`mail/notifier.rb`	Mail::Notifier

Example 10.1. Skeleton scheduled job class (`mail/notifier.rb`)

```
module Mail
  class Notifier

    # implementation goes here

  end
end
```

Each job class should implement a no-argument `run()` method to perform the work when fired. The class may optionally implement a one-argument `on_error(exception)` method to handle any errors raised during the job's execution.

Example 10.2. Scheduled job implementation (`mail/notifier.rb`)

```
module Mail
  class Notifier

    # optional, only needed if you pass config options to the job
    def initialize(options = {})
      @options = options
```

```
    end

    def run()
      # perform work here
    end

    def on_error(exception)
      # Optionally implement this method to interrogate any exceptions
      # raised inside the job's run method.
    end

  end
end
```

From within the class's `run()` method, the full application environment is available.

3. Scheduling Jobs

The job schedule defines the time(s) that a job should execute. This may be defined to be single point in time, or more often, as recurring event. The job schedule is defined in your deployment descriptor.

3.1. Configuration Format

Within the internal `torquebox.yml` descriptor (or through an external `*-knob.yml` descriptor), scheduled jobs are configured using the `jobs:` section, in which a block of information is provided for each job. The block starts with an arbitrary name for the job. Each block must also define the job class and the schedule specification. Optionally a `timeout`, a `description`, and a `config` may be provided. Providing a `timeout` will cause the job to be interrupted if it runs beyond the timeout period (see Section 3.2, “Timing Out Jobs”). If you provide a `config`, its value will be passed to the initialize method of the job class.

If you are using the DSL (via `torquebox.rb`) in your internal descriptor, each job is defined using the `job` directive, with very similar options to the YAML syntax described above. The DSL does not require a name for each job, unless you intend to share a job class across multiple jobs.

Example 10.3. Example deployment descriptor

Using the YAML syntax:

```
application:
  ..
jobs:
  mail.notifier:
    job: Mail::Notifier
```

```
    cron: '0 */5 * * * ?'
    timeout: 50000 ms
    description: Deliver queued mail notifications
    config:
      foo: bar
```

And via the DSL:

```
TorqueBox.configure do
  ...
  job Mail::Notifier do
    name 'mail.notifier' # optional, unless the job class is used by multiple jobs
    cron '0 */5 * * * ?'
    timeout '5s'
    description 'Deliver queued mail notifications' # optional
    config do
      foo 'bar'
    end
  end
end
```

The cron attribute should contain a typical crontab-like entry. It is composed of 7 fields (6 are required).

Seconds	Minutes	Hours	Day of Month	Month	Day of Week	Year
0-59	0-59	0-23	1-31	1-12 or JAN-DEC	1-7 or SUN-SAT	1970-2099 (optional)

For several fields, you may denote subdivision by using the forward-slash (`/`) character. To execute a job every 5 minutes, `*/5` in the minutes field would specify this condition.

Spans may be indicated using the dash (`-`) character. To execute a job Monday through Friday, `MON-FRI` should be used in the day-of-week field.

Multiple values may be separated using the comma (`,`) character. The specification of `1,15` in the day-of-month field would result in the job firing on the 1st and 15th of each month.

Either day-of-month or day-of-week must be specified using the `?` character, since specifying both is contradictory.

3.2. Timing Out Jobs

To keep jobs from running too long, you can set a `timeout` setting for the job. The format of the timeout is a integer followed by an optional time unit. The available time units are:

- `ms` - milliseconds
- `s` - seconds
- `m` - minutes
- `h` - hours

If no unit is provided, seconds are assumed.

In addition to specifying a timeout parameter, you will also have to implement a `on_timeout` method on your job class that will be called when the timeout occurs. This method is responsible for actually shutting down the job - TorqueBox will not kill the job when the timeout occurs. One approach would be for your job to periodically check a flag while processing, with `on_timeout` setting that flag when called.

Example 10.4. Job with timeout (`mail/notifier.rb`)

```
module Mail
  class Notifier

    # optional, only needed if you pass config options to the job
    def initialize(options = {})
      @options = options
      @timeout = false
    end

    def run()
      notification_list.each do |n|
        raise 'Timeout!' if @timeout
        n.notify
      end
    end

    def on_timeout
      @timeout = true
    end

  end
end
```

3.3. Job Concurrency

Quartz manages its own thread pool for running jobs. By default, this pool contains three threads. If you have more than three jobs executing at the same time, you may want to increase this pool size. You can do so via the `concurrency` setting.

Example 10.5. Setting job concurrency

Using the YAML syntax:

```
application:
  ..
jobs:
  concurrency: 10
  mail.notifier:
    job: Mail::Notifier
    cron: '0 */5 * * * ?'
```

And via the DSL:

```
TorqueBox.configure do
  ...
  options_for :jobs, :concurrency => 10

  job Mail::Notifier do
    name 'mail.notifier' # optional, unless the job class is used by multiple jobs
    cron '0 */5 * * * ?'
  end
end
```

Note that if you are using a bounded runtime pool for the jobs subsystem that is smaller than the concurrency setting, your available concurrency will be limited to the pool size. See Chapter 16, TorqueBox Runtime Pooling for more details.

3.4. Jobs Management at Runtime

In addition to creating jobs defined in the deployment descriptors you can create and remove them at runtime too.

3.4.1. Scheduling Jobs at Runtime

It is possible to create a new job at runtime. You need to use the `schedule` method available in the `TorqueBox::ScheduledJob` module. This method returns `true` if the task is completed or `false` otherwise. There is a default timeout set to 30 seconds meaning that if the job will not be scheduled in the mentioned time the method will finish immediately returning `false`.

Example 10.6. Scheduling a job

```
TorqueBox::ScheduledJob.schedule('JobClassName', '*/10 * * * * ?')
```

This simple execution will create a new scheduled job implemented in the `JobClassName` class and run it every 10 seconds.

```
TorqueBox::ScheduledJob.schedule('JobClassName', '*/10 * * * * ?',
      :name => 'simple.job',
      :description => 'This is a simple job',
      :timeout => '2s',
      :config => {"number" => 1, "text" => "bacon"},
      :singleton => false)
```

This example shows all the available options. Please see the table below for explanation.

The `schedule` method is executed asynchronously and returns a `java.util.concurrent.CountDownLatch` object which can be used to wait for the task completion. If you want to have a synchronous method use the `schedule_sync` method. It will block and return `true` after successful task completion and `false` otherwise.

The job class name and cron expression is required. Additionally the `schedule` method accepts following, optional parameters:

Table 10.1. Job scheduling options

Option	Default	Description
`:name`	"default"	The job name unique across the application.
`:description`		Job description.
`:timeout`	"0s"	The time after the job execution should be interrupted. By default it'll never interrupt the job execution.
`:config`		Data that should be injected to the job constructor.
`:singleton`	true	Flag to determine if the job should be executed on every node in the cluster or only on one node (default).

Every job requires a unique name across the application. By default, if there is no `:name` parameter provided the name will be set to the class name. In case the job class name includes module name, like this: `Module::ClassName`, the job name will be set to `Module.ClassName`.

If you schedule a job with a name of a job already deployed - the old job will be replaced with the new one.

Note that if you schedule a job at runtime it'll not be persisted and is lost after the server restart.

3.4.2. Removing Jobs at Runtime

You can easily remove a scheduled job. To do this use the `remove` method available in the `TorqueBox::ScheduledJob` module. This method returns `true` if the task is completed or `false` otherwise. There is a default timeout set to 30 seconds meaning that if the job will not be removed in the mentioned time the method will finish immediately returning `false`.

The `remove` method is executed asynchronously and returns a `java.util.concurrent.CountDownLatch` object which can be used to wait for the task completion. If you want to have a synchronous method use the `remove_sync` method. It will block and return `true` after successful task completion and `false` otherwise.

Example 10.7. Removing a job

```
TorqueBox::ScheduledJob.remove('simple.job')
```

This example will lookup a job with the 'simple.job' name and remove it.

Note that if you remove a job defined in the deployment descriptor, it'll be started again after server restart.

4. 'At' Jobs

'At' jobs are jobs that use different scheduling mechanism compared to regular scheduled jobs where you define the execution time with cron expressions. In case of 'at' jobs you have more control over the execution of the job.

Example 10.8. Examples of scheduling 'at' jobs

```
# The job will be executed every 200 ms, from now, for the next 10 seconds
TorqueBox::ScheduledJob.at('SimpleJob', :every => 200, :until => Time.now + 10)

# The job will be executed for the first time in 10 seconds (current time + 10
 seconds), then every
# 500 ms, for the next 10 seconds (current time + 20 seconds)
```

```
TorqueBox::ScheduledJob.at('SimpleJob', :at => Time.now + 10, :every => 500, :until
 => Time.now + 20)

# The job will be executed for the first time in 5s and repeated 3 times
# The time between executions is set to 1.5s
TorqueBox::ScheduledJob.at('SimpleJob', :in => 5000, :repeat => 3, :every => 1500)
```

The `at` method is executed asynchronously and returns a `java.util.concurrent.CountDownLatch` object which can be used to wait for the task completion. If you want to have a synchronous method use the `at_sync` method. It will block and return `true` after successful task completion and `false` otherwise.

The first parameter of the `at` method is the class name of the job implementation to execute. The second parameter allows to specify when the job should be executed. Below you can find valid options.

Table 10.2. 'At' job scheduling options

Option	Default	Description
`:at`	Time.now	Specifies when the at job should start firing. Must be a `Time` class. Can't be specified with `:in`.
`:in`		Specifies when the at job should start firing, in ms from now. Can't be specified with `:at`.
`:every`		Specifies the delay interval between at job firings, in ms. If specified without a `:repeat` or `:until`, the job will fire indefinitely.
`:repeat`		Specifies the number of times an at job should repeat beyond its initial firing. Requires `:every` to be provided.
`:until`		Specifies when the at job should stop firing. Must be a `Time` class.
`:name`	job class name	The job name unique across the application.
`:description`		Job description.
`:timeout`	"0s"	The time after the job execution should be interrupted. By default it'll never interrupt the job execution.
`:config`		Data that should be injected to the job constructor.

Option	Default	Description
`:singleton`	true	Flag to determine if the job should be executed on every node in the cluster or only on one node (default).

5. Clustered Jobs

5.1. High Availability Singleton Jobs

TorqueBox supports highly-available singleton jobs. By default, a job only runs on one node in the cluster and if that node goes down or the job fails to run to completion, it is automatically scheduled on a new node.

To use high availability singleton jobs, you must start TorqueBox with a clustered configuration. For example:

```
$ $JBOSS_HOME/bin/standalone.sh --server-config=standalone-ha.xml
```

Alternatively, use the `torquebox` command:

```
$ torquebox run --clustered
```

HA jobs are configured using the `singleton` key in the job specification in your deployment descriptor. Its default value is `true` so you must manually configure it with a value of `false` for the job to run on every node in the cluster.

Example 10.9. Example deployment descriptor

Using the YAML syntax:

```
application:
  ..
jobs:
  mail.notifier:
    job:         Mail::Notifier
    cron:        '0 */5 * * * ?'
    description: Deliver queued mail notifications
    singleton: true
    config:
      foo: bar
```

And via the DSL:

```
TorqueBox.configure do
  ...
  job Mail::Notifier do
    name 'mail.notifier' # optional, unless the job class is used by multiple jobs
    cron '0 */5 * * * ?'
    description 'Deliver queued mail notifications' # optional
    singleton true
    config do
      foo 'bar'
    end
  end
end
```

This is the same deployment descriptor from the example above. Including the `singleton` attribute with a value of `true` is redundant of course, since jobs will only run on a single node when clustered, by default.

5.2. Jobs Running on Every Node

To configure a job to run on every node in a cluster, set `singleton` to `false` in the deployment descriptor.

6. Resource Injection with Jobs

If a job requires access to other resources, such as messaging topics and queues, or Java CDI components these should be injected using the resource injection facilities provided by TorqueBox (see Chapter 12, TorqueBox Resource Injection).

TorqueBox Services

1. What Are Services?

Services are persistent background Ruby daemons deployed and managed by TorqueBox. Common uses for services include connecting to a remote service (IRC bot, Twitter Streaming API client) or starting a server to listen for incoming connections. A service may be deployed as part of a web application or as its own application without any web component. Services have full access to the entire Ruby environment. This means that a service deployed as part of a web application can use the app's database models, for example.

2. Service Classes

Each service maps to exactly one Ruby class that should optionally implement `initialize(Hash)`, `start()` and `stop()` methods which should each return quickly. Typically the `start` method will spawn a new thread to start an event loop or other long-running task.

Example 11.1. Service implementation (`my_service.rb`)

```
class MyService
  def initialize(opts={})
    @name = opts['name']
  end

  def start
    Thread.new { run }
  end

  def stop
    @done = true
  end

  def run
    until @done
      puts "Hello #{@name}"
      sleep(1)
    end
  end
end
```

This example service prints a message every second until stopped.

The service's `start` method will be invoked when the service is deployed, and `stop` will be invoked automatically when the service is undeployed. Thus a convenient hook is provided for cleanly shutting down any threads or other resources used by the service.

3. Deploying Services

Services are deployed by creating a `services` section inside your application's deployment descriptor.

3.1. Configuration Format

Within the internal `torquebox.yml` descriptor (or through an external `*-knob.yml` descriptor), services reside under a `services` key of `torquebox.yml`. Each key underneath `services` is either a unique name for the service or the name of the Ruby class implementing the service. Providing a unique name allows the reuse of the same Ruby class to provide multiple services. If the Ruby class name is not used as the key, it must be provided using the `service` key in the key/value pairs nested underneath the service entry as options for the service. Any value assigned to the `config` key underneath the service entry will be passed in as the parameter to the service's `initialize` method.

If you are using the DSL (via `torquebox.rb`) in your internal descriptor, each service is defined using the `service` directive, with very similar options to the YAML syntax described above. The DSL does not require a name for each job, unless you intend to share a job class across multiple jobs.

Example 11.2. Example deployment descriptor

Using the YAML syntax:

```
services:
  MyService:
    config:
      name: TorqueBox User

  AnotherService:

  ham-machine:
    service: FoodMachine
    config:
      food: ham

  biscuit-machine:
    service: FoodMachine
    config:
      food: biscuit
```

This deploys four services; the first two using the class name as the key: `MyService` which corresponds to the example above and `AnotherService` which doesn't take any initialization parameters. The latter two services reuse the same class, and use a unique name as the key.

And using the DSL:

```
TorqueBox.configure do
  ...
  service MyService do
    config do
      name 'TorqueBox User'
    end
  end

  service AnotherService

  service FoodMachine do
    name 'ham-machine'
    config do
      food 'ham'
    end
  end

  service FoodMachine do
    name 'biscuit-machine'
    config do
      food 'biscuit'
    end
  end
```

Service classes should be placed in a directory that is on the application's load path. For Rails applications, the convention is to put your service classes in `$RAILS_ROOT/app/services/`. For non-Rails applications, the convention is to use `$RAILS_ROOT/services/`. No matter what type of application you have, both directories will automatically be added to the load path.

4. Clustered Services

4.1. High Availability Singleton Services

TorqueBox supports highly-available singleton services. By default, a service only runs on one node in a cluster and if that node fails or the service is interrupted for any reason, it automatically starts on another node.

To use highly-available singleton services, you must start TorqueBox with a clustered configuration. For example:

```
$ $JBOSS_HOME/bin/standalone.sh --server-config=standalone-ha.xml
```

Alternatively, use the `torquebox` command:

```
$ torquebox run --clustered
```

HA services are configured using the `singleton` key in the services section of `torquebox.yml`. Its default value is `true` so you must manually configure it with a value of `false` for the service to run on every node in the cluster.

Example 11.3. Example deployment descriptor

Using the YAML syntax:

```
services:
  MyService:
    config:
      name: TorqueBox User

  AnotherService:

  ham-machine:
    singleton: false
    service: FoodMachine
    config:
      food: ham

  biscuit-machine:
    singleton: false
    service: FoodMachine
    config:
      food: biscuit
```

And using the DSL:

```
TorqueBox.configure do
  ...
  service MyService do
    config do
```

```
    name 'TorqueBox User'
  end
end

service AnotherService

service FoodMachine do
  name 'ham-machine'
  singleton false
  config do
    food 'ham'
  end
end

service FoodMachine do
  name 'biscuit-machine'
  singleton false
  config do
    food 'biscuit'
  end
end
```

This is the same deployment descriptor from the example above but this time `FoodMachine` services are configured to run on all nodes in the cluster. The `MyService` and `AnotherService` services are singletons and will only run on one node in the cluster.

4.2. Services Running on Every Node

To configure your services to run on all nodes in a cluster, set `singleton` to `false` in the deployment descriptor.

5. Resource Injection with Services

If a service requires access to other resources, such as messaging topics and queues, or Java CDI components these should be injected using the resource injection facilities provided by TorqueBox (see Chapter 12, TorqueBox Resource Injection).

TorqueBox Resource Injection

1. What is Resource Injection?

Resource injection is the term given a software architectural strategy that moves the responsibility of finding and connecting components to a container, allowing components to remain simple and testable. Components declare what they need, and when instantiated by the container, the container also satisfies those needs.

What's a resource? A resource may be most any component within your application, ranging from instances of Java classes, to messaging destinations.

2. Basics of Resource Injection

TorqueBox supports injection within the context of jobs, services, messaging-handlers, Stomplets and web applications. To look up a value from the injection registry, use the `TorqueBox.fetch(...)` method.

For instance:

```
class MyService

  def initialize()
    @queue = TorqueBox.fetch('/queues/new-accounts')
  end

  def that_thing()
    TorqueBox.fetch(com.foo.ThatThing)
  end

end
```

3. Injectable Resources

A variety of resources may easily be injected with the `TorqueBox.fetch(...)` method.

CDI Resources. The Java Context and Depedency Injection (CDI) spec defines a method for managing relationships between components. CDI-enabled components may be injected by providing a fully-qualified Java class name to the `TorqueBox.fetch(...)` method. Typically CDI components should be packaged in a JAR archive, and placed in your application's `lib/` or `vendor/jars/` directory.

TorqueBox uses the JBoss Weld implementation of CDI. Please see the Weld website [http://seamframework.org/Weld] for more information.

```
class MyService

  def initialize()
    @java_service = TorqueBox.fetch(com.mycorp.MyJavaService)
  end

end
```

JRuby explicitly supports the simple syntax for common US-centric package names starting with `com`, `org`, `net`, `java`, and javax, amongst others. For other top-level packages based on country codes, such as `pl`, `de`, or `za`, to perform injection you should reference your class through the `Java` ruby package.

```
TorqueBox.fetch(Java::pl.softwaremine.PolishingService)
```

CDI Injection Requires Full Distribution

You must be running the full distribution of TorqueBox on top of JBoss EAP in order to use CDI injection.

Messaging Destinations. Message destinations, such as queues and topics, may be injected into your components. If the argument to `TorqueBox.fetch(...)` includes the string fragment `"/queue"` or `"/topic"`, TorqueBox will look up the relevant `TorqueBox::Messaging::Queue` or `TorqueBox::Messaging::Topic`.

Using injection is the preferred method for obtaining a reference to a destination, to ensure that your job, service or web application relying upon the destination does not begin operation until the destination has been completely provisioned.

```
class MyController < ApplicationController

  def create
    notify_topic = TorqueBox.fetch('/topics/new-accounts')
  end
```

```
end
```

Naming & Directory Entries. Arbitrary items within the application's naming environment may be injected if the argument to `TorqueBox.fetch(...)` begins with `"java:comp/env"`.

```
class MyController < ApplicationController

  def create
    jndi_item = TorqueBox.fetch('java:comp/env/that_thing')
  end

end
```

JBoss MSC Services. JBoss Modular Service Container is the container that drives the entire TorqueBox AS. Many components are accessible as MSC Services. These may be injected by passing the ServiceName as a string to `TorqueBox.fetch(...)`.

```
class MyController < ApplicationController

  def create
    the_actual_webserver = TorqueBox.fetch('jboss.web')
  end

end
```

Services. TorqueBox Services may be injected into your components if the argument to `TorqueBox.fetch(...)` begins with `"service:"` followed by the key used to configure the service in `torquebox.yml`.

```
class MyController < ApplicationController

  def stop
    # Service defined with a unique name in torquebox.yml
    the_torque_service = TorqueBox.fetch('service:my_torque_service')
    # Service defined with service class in torquebox.yml
    another_service = TorqueBox.fetch('service:AnotherSerice')
  end

end
```

4. Internals and Testing

At runtime, each `TorqueBox.fetch(...)` method looks up the injected resource through the `TorqueBox::Registry` singleton. In test environments, you may desire to populate this registry, using the `merge!(...)` method, which accepts a key/value `Hash`.

The key for each entry should match either the string argument used with `TorqueBox.fetch(...)`, or the Ruby version of the Java class name, if performing CDI injection. The value should be an appropriate object.

For instance, the Java class of `java.util.Set` should be converted into a string of `"Java::JavaUtil::Set"` when used as an injection look-up key.

TorqueBox Authentication

TorqueBox provides a simple Ruby interface to the underlying JAAS security framework built into the JBoss Application Server. JAAS (Java Authentication and Authorization Service) is a pluggable security framework which intends to shield application developers from the underlying security implementation. We kept with this approach for TorqueBox and have hidden most all of the implementation details so you can focus on writing your applications.

TorqueBox applications can authenticate against any security realm that you have specified in your JBoss configuration file (typically standalone.xml or standalone-ha.xml). configuration. To learn more about how JBoss security works and is configured, refer to the JBoss documentation [https://docs.jboss.org/author/display/AS7/Admin+Guide#AdminGuide-SecurityRealms]. The TorqueBox integration, however, makes authenticating against a corporate JAAS data store trivial.

1. Security Domains

The JBoss Application Server allows application developers to authenticate against any of the JAAS security policies configured in the AS. In addition, TorqueBox adds TorqueBox-specific security policies to the AS when your application is deployed. We refer to these JAAS policy names as "domains". TorqueBox ships with a simple authentication domain, named `torquebox`. The `torquebox` domain uses a `SimpleServerLoginModule` for authentication.

The `SimpleServerLoginModule` login algorithm is: if password is null, authenticate the user and assign an identity of "guest" and a role of "guest". else if password is equal to the user name, assign an identity equal to the username and both "user" and "guest" roles else authentication fails.

To use the `torquebox` domain, specify this in your deployment descriptor:

Example 13.1. Using the `torquebox` domain

Using the YAML syntax:

```
auth:
  default:
    domain: torquebox
```

And via the DSL:

```
TorqueBox.configure do
  ...
```

```
  authentication :default, :domain => 'torquebox'
end
```

The `torquebox` domain is deployed on demand only if your application specifies it in the configuration file. However, note that JAAS security domains are available to all applications deployed within the AS.

In addition to the `torquebox` security domain, an application specific domain - `torquebox-appname` is initialized when your application is deployed. The name of the application is determined from the name of your external descriptor (your `*-knob.yml` file) - the `-knob.yml` is dropped, leaving the application name.

This domain allows you to specify username/password pairs inside your deployment descriptor. Users are authenticated against whatever usernames and passwords you have configured.

Example 13.2. Using the `torquebox` domain

Using the YAML syntax:

```
auth:
  default:
    domain: torquebox-myapp
    credentials:
      john: johnspassword
      alice: alicespassword
```

And via the DSL:

```
TorqueBox.configure do
  ...
  authentication :default do
    domain 'torquebox'
    credential 'john', 'johnspassword'
    credential 'alice', 'alicespassword'
  end
end
```

2. Configuration

TorqueBox authentication is configured in the `torquebox.yml` file or in a separate `auth.yml` by adding an `auth` section. Within this, you may add one or more named authentication handles. For example, let's say your application is a dashboard which allows users to access JMX and HornetQ data. Most of the time, you're going to be using the hornetq domain, but on occasion, you'll want to authenticate against the JMX domain. You can do this within Ruby code by configuring your `auth` section.

When using the DSL in `torquebox.rb`, each authentication entry is specified using the `authentication` directive.

Example 13.3. Using the `torquebox` domain

Using the YAML syntax:

```
auth:
  default:
    domain: hornetq
  jmx:
    domain: jmx-console
```

And via the DSL:

```
TorqueBox.configure do
  ...
  authentication :default, :domain => 'hornetq'
  authentication :jmx, :domain => 'jmx-console'
end
```

A handle to the HornetQ authentication domain is now available to you with:

```
authenticator = TorqueBox::Authentication.default
```

and the JMX authentication domain can be obtained with:

```
authenticator = TorqueBox::Authentication['jmx']
```

3. Ruby API

The Ruby API has 3 methods:

- `default`
- `[]( name )`
- `authenticate( username, password )`

The first two methods, `default` and `[]` are used to get the default authentication domain or to look up an authenticator by name. The last is to actually authenticate a user. To use the Ruby API, require `torquebox` and `torquebox-authentication` as shown below. This code shows a simple Ruby authentication module that authenticates agains the JAAS security configuration.

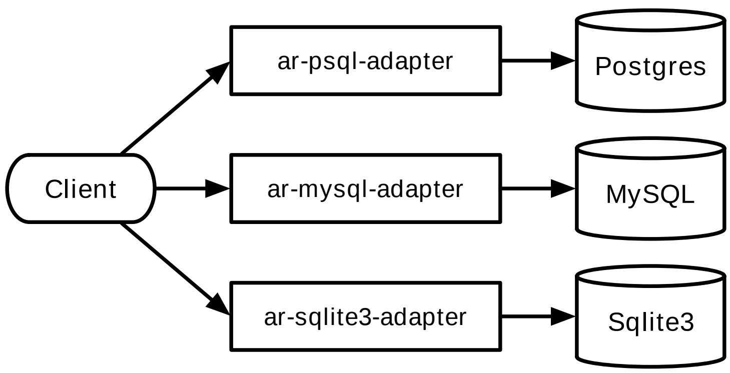
```
require 'torquebox'
require 'torquebox-security'

module MyApp
  module Authentication

    def login_path
      "/login"
    end

    def authenticated?
      !session[:user].nil?
    end

    def authenticate(username, password)
      return false if username.blank? || password.blank?
      authenticator = TorqueBox::Authentication.default
      authenticator.authenticate(username, password) do
        session[:user] = username
      end
    end

    def require_authentication
      return if authenticated?
      redirect login_path
    end

    def logout
      session[:user] = nil
      redirect login_path
    end

  end
end
```

The `authenticate` method accepts a block, allowing you to execute code within an authenticated context.

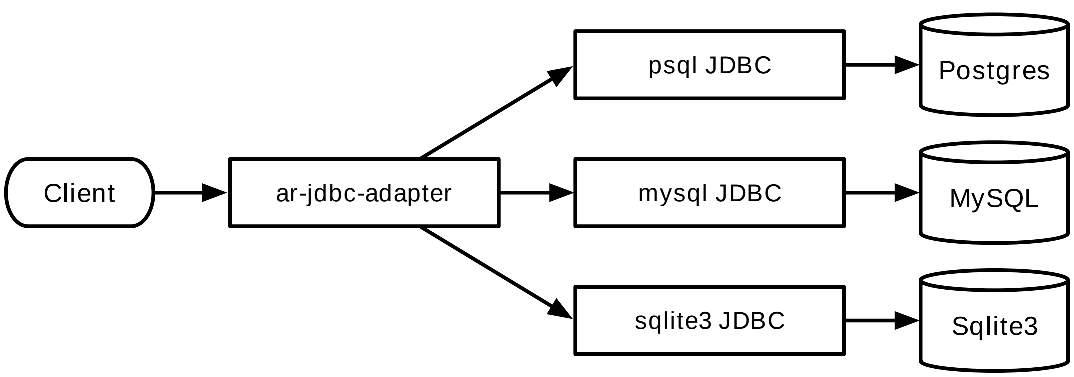
Database Connectivity in TorqueBox

1. ActiveRecord

Typical applications require the use of databases. Within the Rails community, ActiveRecord is one of the more popular database connectivity libraries. With traditional Ruby-based applications, you needed to require the correct ActiveRecord adapter for the database you were connecting to. Each adapter managed the communication between the client and the end database, directly mediating the connection.



Since TorqueBox is based on the JBoss Java environment, it has the capability to use enterprise-grade JDBC (Java Database Connectivity API) drivers. Rails applications can take advantage of these drivers by using the generic ActiveRecord JDBC adapter. The adapter will locate and activate the correct underlying Java JDBC adapter for the target database.



The most visible change required of applications using the JDBC-based ActiveRecord adapter involves the gems your application must rely on. Primarily you must rely on the `activerecord-jdbc-adapter`. This adapter is adjusts ActiveRecord configuration to use the JDBC version of any specified driver.

Additional gems need to be available to your system, depending on your target database:

- `jdbc-postgres`
- `jdbc-mysql`
- `jdbc-sqlite3`

These gems simply embody the Java JAR holding the actual underlying JDBC driver.

No changes to your application's database configuration is required. You still specify the correct driver name for the database, such as `postgresql` or `sqlite3`.

2. DataMapper

Not everyone uses ActiveRecord to connect to a database. TorqueBox also works well with DataMapper, and you don't have to do anything special. A Gemfile for an application which uses DataMapper to connect to a PostgreSQL database looks like this.

```
gem 'data_mapper', '~>1.1'
gem 'dm-core', '~>1.1'
gem 'dm-postgres-adapter', '~>1.1'
gem 'dm-migrations', '~>1.1'
gem 'dm-timestamps', '~>1.1'
gem 'dm-observer', '~>1.1'
```

Initializing DataMapper is unchanged.

```
DataMapper.setup(:default, 'postgres://user:pass@localhost/databasename')
```

3. Raw JDBC

It is also possible to use JDBC directly without the need for a object-relational mapping library. For more information on this, see the JRuby wiki [https://github.com/jruby/jruby/wiki/JDBC].

4. Distributed Transactions

TorqueBox includes support for distributed (XA) transactions. Depending on your database, you may need to alter its configuration to take advantage of transactions. See Chapter 15, TorqueBox Distributed Transactions for more details.

TorqueBox Distributed Transactions

1. Overview

TorqueBox takes advantage of its host's robust transactional facilities. JBoss provides state-of-the-art distributed XA transaction support, and TorqueBox exposes this to Ruby developers in a concise, often transparent API.

It's important to understand the difference between a conventional database transaction and a distributed transaction: multiple resources may participate in a distributed transaction. The most common example of a transactional resource is a relational database, but other examples include message brokers and some NoSQL data grids. Distributed transactions allow your application to say, tie the success of a database update to the delivery of a message, i.e. the message is only sent if the database update succeeds, and vice versa. If either fails, both rollback.

In addition, Rails `ActiveRecord` models are enhanced when run in TorqueBox so that connections from multiple, class-specific databases can indeed participate in a single distributed transaction. Further, the behavior of nested transaction rollbacks won't surprise you: if the child rolls back, the parent will, too, excepting when the `:requires_new` option is passed to the child. Callbacks for `after_commit` and `after_rollback` work as one would expect.

2. The `TorqueBox.transaction` method

You may explicitly demarcate a transaction using `TorqueBox.transaction`. If the block of commands you pass to it runs to completion without raising an exception, the transaction is committed. Otherwise, it is rolled back. It's just that simple. It accepts the following arguments:

- An arbitrary number of XAResources to enlist in the current transaction. This is rarely needed since TorqueBox message destinations, background tasks and caches are all transactionally aware. They will enlist themselves in the transaction defined by `TorqueBox.transaction` automatically, by default.
- Optionally, either a symbol or a hash indicating the scope of the transaction. The `:scope` attribute provides analogs to the JEE transaction attributes [http://docs.oracle.com/javaee/6/tutorial/doc/bncij.html]. The default is `:required`.
- A block defining your transaction. All actions taken in the block will be committed after the block is called unless an exception is raised, in which case the transaction will be rolled back.

Table 15.1. Transaction Scopes

Scope	Description
`:required`	Execute within current transaction, if any, otherwise start a new one, execute, commit or rollback.

Scope	Description
:requires_new	Suspend current transaction, if any, start a new one, execute, commit or rollback, and resume the suspended one.
:not_supported	Suspend current transaction, if any, and execute without a transaction. Also, :none is an alias.
:supports	Execute the body whether there's a transaction or not; may lead to unpredictable results
:mandatory	Toss an exception if there's no active transaction
:never	Toss an exception if there is an active transaction

Example 15.1. Nesting transactions with different scopes

```
TorqueBox.transaction do
  # ... tx #1 created due to default :required scope
  TorqueBox.transaction(:none) do
    # ... tx #1 suspended
    # exceptions raised here won't rollback actions in this block
    TorqueBox.transaction(:scope => :requires_new) do
      # ... tx #2 begun
      # exceptions raised here will rollback tx #2
    end
    # ... tx #2 committed
  end
  # ... tx #1 resumed
  TorqueBox.transaction(:mandatory) do
    # ... actions become a part of tx #1
  end
  TorqueBox.transaction(:requires_new => true) do  # deprecated syntax
    # ... tx #1 suspended and tx #3 begun
  end
  # ... tx #3 committed (or rolled back) and tx #1 resumed
  # exceptions raised (or uncaught) here will rollback tx #1
end
# ... tx #1 committed
```

Obviously, the above example is contrived. When multiple transactional components collaborate, you don't often know how methods, invoked directly or indirectly, might demarcate their transactions. Rarely would you explicitly nest transactions within one method, but the above serves as an example showing the effects of transaction scope.

The above example also shows the options for setting scope either as a symbol, e.g. `:requires_new`, or a hash, e.g. `:scope => :requires_new`. The deprecated syntax, `:requires_new => true`, matching the Rails convention, is provided for backwards compatibility.

3. Messaging

By default, `MessageProcessors` are not transactional. To make them transactional, add `xa: true` to the processor's entry in `torquebox.yml` or `torquebox.rb` - see Section 4.8.3, “Connecting Consumers to Destinations” for more information.

After you've enabled transactions for the message processor, each `on_message(msg)` invocation demarcates a transaction. If no exceptions are raised, the transaction commits. Otherwise, it rolls back. This is the default behavior and requires no additional configuration on your part.

Any messages published to any JMS destinations automatically become part of the current transaction, by default. So they won't be delivered until that transaction commits.

Any manipulations of your Rails `ActiveRecord` models (persisted to your XA-compliant database) within `on_message(msg)` will become part of its transaction if distributed transactions have been enabled for that database.

Occasionally, you may not want a published message to assume the active transaction. In that case, pass `:tx => false`, and the message will be delivered whether the active transaction commits or not. This option works for backgrounded tasks as well.

`Backgroundable` tasks are not transactional.

4. Database Configuration

Ensure your application is correctly configured to use the `activerecord-jdbc-adapter` and then add `xa: true` to your `database.yml` entry for each database you want to use in a distributed transaction. See Chapter 14, Database Connectivity in TorqueBox for more details on configuring ActiveRecord JDBC support.

Distributed transactions are restricted to those databases supported by both the `activerecord-jdbc-adapter` and JBoss XA datasources. Currently, that includes PostgreSQL, MySQL, H2, Derby, Oracle, Microsoft SQL Server, and IBM DB2. Sqlite3 doesn't support XA. Default installations of some of these databases may require additional configuration to support XA.

4.1. PostgreSQL

To enable full distributed transaction support in PostgreSQL, you'll need to set `max_prepared_transactions` to something greater than zero in `postgresql.conf`, which is the usual default in most installations. Changing it requires a server restart.

4.2. MySQL

To achieve transactional support -- even non-distributed functionality -- you must enable the `InnoDB` storage engine. As of MySQL 5.5, this is the default storage engine.

4.3. Example

Example 15.2. Enabling XA support (`config/database.yml`)

```
production:
  adapter: mysql
  xa: true
  database: my_database
  host: my_host
  username: my_username
  password: my_password
  encoding: utf8
```

TorqueBox Runtime Pooling

To run Ruby code inside a Java application server, the TorqueBox platform requires a Ruby interpreter, provided by JRuby [http://www.jruby.org]. TorqueBox provides a simple but flexible means of mapping the app server's threads of execution to one or more Ruby interpreters, giving you complete concurrency control, but the defaults should be reasonable.

1. Types of Runtime Pools

TorqueBox defines two types of pools from which a Ruby interpreter may be obtained:

- Bounded
- Shared

Bounded pools. A bounded pool is a typical resource pool with minimum and maximum capacity. Each interpreter managed by the pool is given out to a single client at a time. It is unavailable for any other client until the current owner returns it to the pool. The pool will ensure that a minimum number of interpreters are kept in the pool at all times. Additionally, a maximum capacity is specified to ensure that the pool does not grow unbounded. Clients requesting an interpreter from a pool with no available interpreters will block until an interpreter becomes available. Interpreters may become available through other clients returning an existing interpreter, or by the pool spinning up additional interpreters, if it has not reached its maximum capacity.

Shared pools. A shared pool is a false pool. A shared pool contains one Ruby interpreter that is allowed to be shared, concurrently, with an unbounded number of clients. A shared pool may only be used in cases where the application is considered threadsafe. An application's threadsafety may be affected by both framework code and deployment factors. These issues are discussed below.

2. Subsystem Pools

As noted above, an advanced application may use the functionality of multiple subsystems. Each subsystem is configured to use a distinct pool in order to provide a modicum of isolation and prevent wayward interaction. The configuration of various subsystem pools are affected by how the application is deployed. Each subsystem is automatically configured using reasonable defaults, but may be completely configured manually through a deployment descriptor (see Chapter 5, TorqueBox Deployment Descriptors).

Web (Rack). The web subsystem, powering Rack applications, defaults to deploying a shared pool. Modern frameworks have mostly moved away from their assumption of single-threaded applications. By using a shared pool, resources are conserved, and a single Ruby interpreter may handle all requests from web clients.

Scheduled Jobs. The pool deployed for the scheduled jobs subsystem varies based on the deployment mode of the application. In `development` mode, automatic code-reloading is desirable, but multiple jobs executing and/or resetting the application within a single interpreter causes race conditions and poor interactions. For this reason, a non-shared bounded pool is configured when the application is deployed in `development` mode. In non-`development` deployments, reloading is disabled, and the race conditions do not exist. In the non-development cases, a more efficient shared pool is configured for the application.

Message Processors. As with the jobs subsystem, asynchronous message processing introduces race conditions between processors executing and processors attempted to reset the application. Likewise, the pool for the message processor subsystem uses a bounded pool when the application is deployed in development mode, otherwise it uses the more efficient shared pool strategy.

3. Configuration

If your application is not designed to be thread-safe, you can instead pool the interpreters resulting in a single-threaded model. You can do this for jobs, messaging, and/or web requests. Typically, if your application creates and uses global variables to manage state for a single web request, you may have problems with the default multi-threaded behavior.

To modify the default interpreter pool configuration, you can add pooling: section to either your application's internal deployment descriptor, or through an external `*-knob.yml` descriptor. This section is always optional, and only required if you wish to modify the defaults.

3.1. Syntax

Within a deployment descriptor, a block may be added for each susbsystem you desire to explicitly configure. Any subsystem not mentioned will be configured with its defaults. Configuration of each type of pool is slightly different.

Subsystem	Key
Web/Rack	`web`
Scheduled jobs	`jobs`
Message Processors	`messaging`
Services	`services`
Stomplets	`stomplets`

Bounded pools. A bounded pool has a `type` parameter of 'bounded' and requires two additional parameters: `min` and `max`. The `min` parameter specifies the minimum number of managed interpreters that pool should initialize itself with. The `max` parameter specifies the largest capacity the pool should ever grow to in order to satisfy client requests.

Example 16.1. Configuring a bounded pool

Using the YAML syntax:

```
pooling:
  web:
    type: bounded
    min: 3
    max: 10
```

And via the DSL:

```
TorqueBox.configure do
  ...
  pool :web do
    type :bounded
    min 3
    max 10
  end
end
```

Shared pools. A shared pool requires no configuration other than indicating a subsystem should use a shared pool.

Example 16.2. Configuring a shared pool

Using the YAML syntax:

```
pooling:
  web:
    type: shared
```

And via the DSL:

```
TorqueBox.configure do
  ...
  pool :web do
    type :shared
  end
end
```

Lazy pools. A lazy pool is a bounded or shared pool that does not start until it is needed. So, a lazy messaging pool would not start until the first message was received. A lazy web pool would not start until the first web request cames in. An eager pool (opposite of lazy), on the other hand, starts when the application is deployed even if it isn't needed yet.

Example 16.3. Configuring eager and lazy pools

Using the YAML syntax:

```
pooling:
  web:
    lazy: false
  jobs:
    lazy: true
```

And via the DSL:

```
TorqueBox.configure do
  ...
  pool :web do
    lazy false
  end
  pool :jobs do
    lazy true
  end
end
```

3.2. Examples

Example 16.4. Default development-mode pooling

Using the YAML syntax:

```
application:
  ...

pooling:
  jobs:
    type: bounded
    min: 1
    max: 2
    lazy: true
  messaging:
    type: bounded
```

```
    min: 1
    max: 2
    lazy: true
  web:
    type: shared
    lazy: false
```

And via the DSL:

```
TorqueBox.configure do
  ...
  pool :jobs do
    type :bounded
    min 1
    max 2
    lazy true
  end

  pool :messaging do
    type :bounded
    min 1
    max 2
    lazy true
  end

  pool :web, :type => :shared, :lazy => false
end
```

Above is the implicit default configuration for an application deployed in `development` mode.

Example 16.5. Default non-development-mode pooling

Using the YAML syntax:

```
application:
  ...

pooling:
  jobs:
    type: shared
    lazy: true
  messaging:
    type: shared
```

```
    lazy: true
  web:
    type: shared
    lazy: false
```

And via the DSL:

```
TorqueBox.configure do
  ...
  pool :jobs, :type => :shared, :lazy => true
  pool :messaging, :type => :shared, :lazy => true
  pool :web, :type => :shared, :lazy => false
end
```

Above is the implicit default configuration for an application deployed in a mode other than `development`.

4. Runtime Initialization

Ruby runtimes are initialized with standard load paths, such as `./lib` enabled. Rails applications benefit from the load path magic that Rails performs automatically. For non-rails applications, TorqueBox initializes Ruby runtimes with `./lib` and `./config` added to the load path. Additionally, for custom runtime initialization, you may place a `torquebox_init.rb` in `./config` or the root directory of your application. This file will be evaluated for all runtimes as they are initialized.

The torquebox Command

When you install TorqueBox you get a `torquebox` command line utility that can be used to deploy and undeploy applications, start and stop the server, and more. Running `torquebox` without any arguments displays the help screen.

```
$ torquebox
Tasks:
  torquebox archive ROOT          # Create a nice self-contained
 application...
  torquebox cli                   # Run the JBoss AS7 CLI
  torquebox deploy ROOT           # Deploy an application to TorqueBox
  torquebox env [VARIABLE]        # Display TorqueBox environment variables
  torquebox exec [KNOB] [COMMAND] # Execute a command within the context
 of ...
  torquebox help [TASK]           # Describe available tasks or one
 specific...
  torquebox list                  # List applications deployed to
 TorqueBox ...
  torquebox rails ROOT            # Create a Rails application at ROOT
 using...
  torquebox run                   # Run TorqueBox (binds to localhost, use
 -...
  torquebox undeploy ROOT         # Undeploy an application from TorqueBox
```

1. torquebox deploy

Running `torquebox deploy` will deploy your current working directory as an application to TorqueBox. If provided with a ROOT path, such as `torquebox deploy /path/to/my/app` the command will deploy the application found at that path.

```
$ torquebox deploy myapp
Deployed: myapp-knob.yml
    into: /opt/torquebox/jboss/standalone/deployments
```

ROOT can be directory containing the application you want to deploy, a -knob.yml file, a .knob archive, or any Java deployable artifact (.war, .ear, etc).

Table 17.1. torquebox deploy options

Option	Description
`--context-path=CONTEXT_PATH`	The web context path for your application. (e.g. / or /myapp)
`--env=ENV`	The application environment. (e.g. development, test, production)
`--name=NAME`	The name of the deployment artifact. (e.g. myapp)

2. torquebox undeploy

Just the opposite of `torquebox deploy` is `torquebox undeploy [ROOT]`. This command will undeploy your application from TorqueBox. Similar to deploying, running this with no arguments, will attempt to undeploy an application with the same name as the current working directory. Providing a name or path, will cause `torquebox undeploy` to attempt to undeploy an application with that name.

Table 17.2. torquebox undeploy options

Option	Description
`--name=NAME`	The name of the deployment artifact to undeploy. (e.g. myapp)

```
$ torquebox undeploy myapp
Attempting to undeploy myapp-knob.yml
Undeployed: myapp-knob.yml
      from: /opt/torquebox/jboss/standalone/deployments
```

3. torquebox run

The `torquebox run` command will run the TorqueBox server. In development, this is similar to `rails server`.

Table 17.3. torquebox run options

Option	Description
`--clustered`	Runs TorqueBox in clustered mode.
`--data-directory=DATA-DIRECTORY`	Override the directory TorqueBox uses to store it runtime data.
`--extra=EXTRA`	Extra options to pass through to JBoss AS, you will to escape dashes with \ (e.g. \--help)
`--max-threads=N`	Maximum number of HTTP threads

Option	Description
`--bind-address=BIND-ADDRESS`	IP address to bind to - don't set this to 0.0.0.0 if used with --clustered
`--node-name=NODE-NAME`	Override the name of the node (which by default is the hostname)
`--port-offset=N`	Offset all port numbers listened on by TorqueBox by this number
`--jvm-options=JVM-OPTIONS`	Options to be passed to the JVM

The server will retain control of the console while it is running. To stop the server, simply send a `SIGINT`, typically by typing control-C.

4. torquebox rails

The `torquebox rails [ROOT]` command creates a Rails application at ROOT using the TorqueBox Rails template, or applies the TorqueBox template to an existing Rails application at ROOT. As with other commands that take a ROOT argument, if ROOT is omitted, the command will operate on the current working directory.

```
$ torquebox rails /path/to/my/app # apply the TorqueBox Rails template to an
 app
```

```
$ torquebox rails # Create a new Rails application in the current directory
```

5. torquebox archive

The `torquebox archive [ROOT]` command creates an application archive containing all of your application dependencies. The archive can be deployed to TorqueBox with the `--deploy` option or by hand after the archive file, known as a .knob, has been created by using `torquebox deploy myapp.knob`. If ROOT is omitted, the command will operate on the current directory.

Table 17.4. torquebox archive options

Option	Description
`--deploy`	Deploys the resulting archive to TORQUEBOX_HOME
`--package_gems`	Include all Bundler gem dependencies in the archive.
`--package_without=GROUPS`	Package without these Bundler groups.
`--precompile_assets`	Precompile all assets (Rails-specific).

6. torquebox cli

The `torquebox cli` command runs the JBoss AS7 command line interface.

```
$ torquebox cli
```

7. torquebox env

The `torquebox env [VARIABLE]` command displays the TorqueBox environment variables TORQUEBOX_HOME, JBOSS_HOME and JRUBY_HOME. The optional `VARIABLE` argument can be one of these three values and will cause the command to display only that value.

```
$ torquebox env
TORQUEBOX_HOME=/opt/torquebox
JBOSS_HOME=/opt/torquebox/jboss
JRUBY_HOME=/opt/torquebox/jruby
```

```
$ torquebox env JBOSS_HOME
/opt/torquebox/jboss
```

8. torquebox list

The `torquebox list` command displays all applications currently deployed to torquebox and their deployement status (e.g. deployed, awaiting deployment, failed). that value.

```
$ torquebox list
cachetest
  Descriptor: /opt/torquebox/jboss/standalone/deployments/cachetest-knob.yml
  Status: deployed

$ torquebox deploy
Deployed: myapp-knob.yml
    into: /opt/torquebox/jboss/standalone/deployments

$ torquebox list
cachetest
  Descriptor: /opt/torquebox/jboss/standalone/deployments/cachetest-knob.yml
  Status: deployed
```

```
myapp
  Descriptor: /opt/torquebox/jboss/standalone/deployments/myapp-knob.yml
  Status: awaiting deployment
```

9. torquebox exec

The `torquebox exec [KNOB] [COMMAND]` command executes the given command within the context of a TorqueBox application, given as the path to a Knob file. A common example might be:

```
$ torquebox exec /path/to/myapp.knob 'rake db:migrate RAILS_ENV=production'
```

This can also be run from the non-gem installation of TorqueBox in this form:

```
$ $TORQUEBOX_HOME/jruby/bin/jruby -S torquebox exec myapp.knob 'rake -T'
```

In both cases, this command assumes that the Knob file has been built with the `--package_gems` option, or that the necessary gems are available to the JRuby runtime in some other way.

Table 17.5. torquebox exec options

Option	Description
`--no_bundle`	Run the given command without `bundle exec`

TorqueBox Rake Support

1. Overview

TorqueBox includes a support package which includes Rake tasks which assist in the deployment to and undeployment from an instance of the TorqueBox Server, in addition to the launching of the server. This rake-based support is normally intended for development-time usage, and not for production. More advanced tooling, such as Capistrano (see Capistrano Support) is advisable for production environments.

First, the `$TORQUEBOX_HOME` and `$JBOSS_HOME` variables must be set to the path of the top of your TorqueBox Installation and the JBoss installation inside of it, respectively, as described in Chapter 2, TorqueBox Installation.

```
$ export TORQUEBOX_HOME=/path/to/torquebox
$ export JBOSS_HOME=$TORQUEBOX_HOME/jboss
```

To include these tasks into your `Rakefile`, use a `single` require statement.

```
require 'torquebox-rake-support'
```

Once these variables are set and you have adjusted your Rakefile, you may perform directory- or archive-based deployments and control the execution of the TorqueBox AS.

2. Deploying applications

2.1. Directory-based deployments

The typical usage of the rake tasks is to perform a deployment of your current application into a local TorqueBox AS during development. The simplest deployment form will deploy the application with `RACK_ENV` or `RAILS_ENV` set to `development`, no virtual host, at the root of the server.

```
$ rake torquebox:deploy
```

If you wish to deploy with a different value for RACK_ENV or RAIL_ENV, the task respects your current shell's values for those variables.

```
$ RAILS_ENV=staging rake torquebox:deploy
```

You may supply a name argument, either as a rake parameter or as an environment variable, to adjust the name of your -knob.yml file. If not supplied, the name of the deployment defaults to the current directory name.

```
$ rake torquebox:deploy['/my-app','foo']
```

```
$ rake torquebox:deploy NAME=foo
```

For example, running "rake torquebox:deploy NAME=foo" will create a deployment artifact called "foo-knob.yml" and deploy it accordingly.

Additionally, a custom context path may be used instead of the default to of `/`, by providing a rake argument to the `torquebox:deploy` task.

```
$ rake torquebox:deploy['/my-app']
```

2.2. Archive-based deployments

In the event you need to deploy the application as an archive, instead of as a directory of loose files, the rake support includes a task to do just that. Additional, the rake task may also be used to simply create the archive without deploying it, if you intend to distribute it to your servers in some other fashion.

To create (but not deploy) an archive:

```
$ rake torquebox:archive
```

Additionally, you can specify a name for the archive, either on the command line or as an environment variable. For example, either of these statements:

```
$ rake torquebox:archive[baz]
```

```
$ rake torquebox:archive NAME=baz
```

will produce an archive called "baz.knob".

The resulting archive will be placed at the root of the application, with a suffix of `.knob`. To inspect the contents, you may use the `jar` tool.

```
$ jar tf myapp.knob
META-INF/
META-INF/MANIFEST.MF
```

```
app/
app/controllers/
app/controllers/application_controller.rb
...
```

You may also have the archive deployed immediately after creating it, in a single command. Here, as before, you may specify a name for the archive.

```
$ rake torquebox:deploy:archive
```

```
$ rake torquebox:deploy:archive[baz]
```

```
$ rake torquebox:deploy:archive NAME=baz
```

3. Undeploying applications

To undeploy an application, either a directory- or archive-based deployment, a single command may be used:

```
$ rake torquebox:undeploy
```

...but we also support torquebox:undeploy:archive for symmetry's sake:

```
$ rake torquebox:undeploy:archive
```

```
$ rake torquebox:undeploy:archive[baz]
```

```
$ rake torquebox:undeploy:archive NAME=baz
```

4. Server control

TorqueBox provides rake tasks for controlling the server.

```
$ cd $TORQUEBOX_HOME; jruby -S rake -T
(in /opt/torquebox)
rake torquebox:check              # Check your installation of the TorqueBox ...
rake torquebox:run                # Run TorqueBox server
rake torquebox:upstart:check      # Check if TorqueBox is installed as an ups...
rake torquebox:upstart:install    # Install TorqueBox as an upstart service
rake torquebox:upstart:restart    # Restart TorqueBox when running as an upst...
```

```
rake torquebox:upstart:start    # Start TorqueBox when running as an upstar...
rake torquebox:upstart:stop     # Stop TorqueBox when running as an upstart...
```

- `torquebox:check`: Check your TorqueBox installation
- `torquebox:run`: Run TorqueBox

  The server will retain control of the console while it is running. To stop the server, simply send a `SIGINT`, typically by typing control-C.
- `torquebox:upstart:check`: Check if TorqueBox is installed as an upstart service
- `torquebox:upstart:install`: Install TorqueBox as an upstart service
- `torquebox:upstart:restart`: Restart TorqueBox when it is running as an upstart service
- `torquebox:upstart:start`: Start TorqueBox when it is installed as an upstart service
- `torquebox:upstart:stop`: Stop TorqueBox when it is installed as an upstart service

Note: The `upstart:install` task makes a couple of assumptions you need to take into account.

- You must have a 'torquebox' user on your system.
- The rake task attempts to create a symlink from $TORQUEBOX_HOME to /opt/torquebox. Run the task as a user with sufficient permissions so that this does not fail.

TorqueBox Capistrano Support

1. What is Capistrano?

Capistrano is a deployment tool to assist in moving code from a repository to a production server. It's a set of tools used from one machine (the deployer), to get an application running on a remote machine (the server).

In many cases, the deployer is a developer working from his or her laptop. Capistrano is installed here. The deployer invokes the tooling locally on his laptop, and Capistrano reaches across the network to set up the right version of the application and activate it within TorqueBox.

2. Installing Capistrano

The TorqueBox distribution includes support for Capistrano, but does not include Capistrano itself. Capistrano requires a few other gems in order to function effectively. It is easy to install everything.

```
$ jruby -S gem install jruby-openssl ffi-ncurses capistrano
```

3. Capify your Application

You can skip this section if you're already using Capistrano with your application. Otherwise, you'll need to `capify` your application to set it up for use with Capistrano.

Ensure that you are in the root of your application's source tree, and run the `capify` command.

```
$ jruby -S capify .
```

This creates a Capfile in the root of your application, which delegates to another file it created: `config/deploy.rb`. The deploy.rb file is the primary location for configuring your deployment strategy.

3.1. Basic deploy.rb configuration

All applications, whether using TorqueBox or another server, require some common settings to be used with Capistrano. The default `deploy.rb` indicates some typical variables you should customize for your deployment.

4. TorqueBox-specific deploy.rb configuration

Within your `deploy.rb`, there are a few additional steps and variables you may configure in order to deploy to a remote TorqueBox server.

4.1. Include TorqueBox recipes

First, you should include the Capistrano recipes which support TorqueBox deployments. If you use Bundler, you should also include the Bundler recipes at this point.

```
require 'torquebox-capistrano-support'
require 'bundler/capistrano'
```

Note: You will need to install the `torquebox-capistrano-support` gem if you are using the `torquebox-server` gem install, as it is not installed by default.

```
$ jruby -S gem install torquebox-capistrano-support
```

4.2. Set up home variable(s)

Capistrano needs to know some details about how TorqueBox is installed on the remote server. Primarily, it needs to be able to locate JBoss and JRuby.

If you've installed TorqueBox by unzipping the distribution, you only need to set `:torquebox_home` in your `deploy.rb`.

```
set :torquebox_home,    '/opt/torquebox/current'
```

If you have a non-standard installation of the TorqueBox components, you may instead set `:jboss_home` and `:jruby_home` individually.

```
set :jboss_home,        '/opt/jboss-as'
set :jruby_home,        '/usr/local/jruby'
```

Capistrano uses these values in order to control the TorqueBox AS process, deploy applications to the correct location, and execute Bundler on the remote server if required. If required, you may also set `:jruby_opts` variable to pass to all invocations of JRuby.

4.3. Optionally configuration application variables

Typical usage of Capistrano expects production values to be embedded into your application's `torquebox.yml` file. In the event you need to override some values when deploying with Capistrano, several application variables may be set. If these are not set, they will not be emitted by Capistrano into the `*-knob.yml` it deploys.

Table 19.1. Application variables

Name	Description
:app_host	String to use a the web virtual host.

Name	Description
:app_context	Application web context.
:app_environment	Hash of name/values for environment variables.
:app_ruby_version	Ruby compatibility version (defaults to 1.9)

4.4. Configure server control style

The TorqueBox AS can be controlled in two different ways. By default, the init.d method is used, but using the bin/ scripts that ship with JBoss is also supported.

init.d. Using a /etc/init.d script, the TorqueBox AS can be integrated into the server's normal service boot sequence and controlled using standard tools and methods enjoyed by sysadmins. By default, Capistrano support assumes the init.d script is located at `/etc/init.d/jbossas`. If you use a differently-named script, simply specify it using the `:jboss_init_script` variable.

```
set :jboss_init_script,    '/etc/init.d/jboss-as7-custom'
```

When using an `init.d` script, it is assumed that other details, such as bind IP address, server configuration selection, and other details are set through `/etc/sysconfig` files.

bin/ scripts. If you do not have access to modify scripts under `/etc/init.d`, you may desire to simply use the `run.sh` and `shutdown.sh` scripts under `$JBOSS_HOME/bin` to control the server process. To enable this method of server control, you must set the `:jboss_control_style` variable.

```
set :jboss_control_style,    :binscripts
```

When using `bin/` scripts, you may control additional server properties through your `deploy.rb` file.

Table 19.2. Variables affecting bin/ script server control

`:jboss_bind_address`	`0.0.0.0`	The IP address to bind when launching the AS.

4.5. Sample deploy.rb File

```
require 'torquebox-capistrano-support'
require 'bundler/capistrano'

# SCM
set :application,       "myapp.com"
set :repository,        "git@github.com:account/repo.git"
set :branch,            "torquebox-2.0"
```

```
set :user,              "torquebox"
set :scm,               :git
set :scm_verbose,       true
set :use_sudo,          false

# Production server
set :deploy_to,         "/opt/apps/myapp.com"
set :torquebox_home,    "/opt/torquebox/current"
set :jboss_init_script, "/etc/init.d/jboss-as-standalone"
set :rails_env,         "production"
set :app_context,       "/"

ssh_options[:forward_agent] = false

role :web, "www.myapp.com"
role :app, "torquebox.myapp.com"
role :db,  "torquebox.myapp.com", :primary => true
```

4.6. Perform deployments

Once your application is setup and configured, and your deployment server is prepared, you can begin performing deployments as you normally would.

Disable the AS. TorqueBox AS can work behind another webserver such as Apache httpd. Capistrano supports placing a `maintenance.html` page to be served by Apache when you desire to take down the app server.

```
$ jruby -S cap deploy:web:disable
```

Capistrano will provide instructions for setting up Apache to stop directing requests to the AS when the maintanence page is in-place. When using TorqueBox behind Apache, these rules normally should live in the <VirtualHost> section of your httpd.conf, instead of within an .htaccess.

```
ErrorDocument 503 /system/maintenance.html
RewriteEngine On
RewriteCond %{REQUEST_URI} !.(css|gif|jpg|png)$
RewriteCond %{DOCUMENT_ROOT}/system/maintenance.html -f
RewriteCond %{SCRIPT_FILENAME} !maintenance.html
RewriteRule ^.*$  -  [redirect=503,last]
```

Deploy the application. The Capistrano deployment workflow can occur even if the TorqueBox AS is not currently running. Deployment will not automatically start the AS if it is not running. Deployment will also never restart the server, as new application deployments are automatically recognized by the running AS.

```
$ jruby -S cap deploy
```

To restart an application but not the server itself, use `restart`

```
$ jruby -S cap deploy:restart
```

Control the TorqueBox AS. You can start and stop the TorqueBox AS independent of deployment activities. When started, all applications that were running when last shutdown will be redeployed.

```
$ jruby -S cap deploy:torquebox:stop
```

To start the TorqueBox AS and re-deploy all previously-running applications:

```
$ jruby -S cap deploy:torquebox:start
```

To restart the TorqueBox AS and re-deploy all previously-running applications:

```
$ jruby -S cap deploy:torquebox:restart
```

torquebox-server Gem

One of the new features is the ability to install TorqueBox as a gem instead of the zip-based installation. The gem installation gives you access to a new `torquebox` command to deploy and undeploy applications and start Torquebox.

1. Install JRuby

Before installing the torquebox-server gem you'll want the latest JRuby installed. We recommend at least 1.6.7 since it fixes an out of memory error during gem install.

Follow instructions at http://jruby.org to install JRuby if it isn't already.

2. Install torquebox-server

```
$ jruby -S gem install torquebox-server
```

If you're using a JRuby version older than 1.6.7, be you'll also need to pass "-J-Xmx1024m" to the jruby command above.

3. Deploying and Undeploying Applications

The torquebox-server gem ships with a `torquebox` binary, which may be used to deploy and undeploy applications, as well as starting the server and other functions. For complete documentation, see Chapter 17, The torquebox Command.

To deploy an application to TorqueBox:

```
$ torquebox deploy /path/to/my_app
```

To undeploy that same application:

```
$ torquebox undeploy /path/to/my_app
```

If you omit a path, the commands default to deploying or undeploying the application in the current directory.

Deployment Help.

```
$ torquebox help deploy
Usage:
  torquebox deploy ROOT
```

```
Options:
  [--context-path=CONTEXT_PATH]  # Context Path (ex: /, /my_app)
  [--env=ENV]                    # Application Environment (ex: development,
 test, production)
  [--name=NAME]                  # The desired name of the deployment artifact
 (ex: foo)

Description:
  Deploy an application to TorqueBox. The ROOT argument should point to either
 a
  directory containing the application you want to deploy, a -knob.yml file, a
  .knob archive, or any Java deployable artifact (.war, .ear, etc).
```

Undeployment Help.

```
$ torquebox help undeploy
Usage:
  torquebox undeploy ROOT

Options:
  [--name=NAME]  # The name of the artifact to undeploy (ex: foo)

Undeploy an application from TorqueBox
```

4. Running

Running TorqueBox is as simple as:

```
$ torquebox run
```

Out of the box, TorqueBox only is only accessible from localhost. To access it from other machines pass the -b parameter to bind to a real IP address or any available IP address:

```
$ torquebox run -b 10.100.10.25
$ torquebox run -b 0.0.0.0
```

To run TorqueBox in clustered mode, use:

```
$ torquebox run --clustered
```

Multiple instances of TorqueBox can run on the same machine. You'll need to pass a unique node name, data directory, and bind each instance to a different IP address or use port offsets. Below are examples of setting up a local two-node cluster using different IP addresses and port offsets, respectively.

```
$ torquebox run --clustered --node-name=node1 --data-directory=/tmp/node1 -b
 10.100.10.25
$ torquebox run --clustered --node-name=node2 --data-directory=/tmp/node2 -b
 10.100.10.26
```

```
$ torquebox run --clustered --node-name=node1 --data-directory=/tmp/node1
$ torquebox run --clustered --node-name=node2 --data-directory=/tmp/node2 --port-
offset=100
```

Run Help.

```
$ torquebox help run
Usage:
  torquebox run

Options:
      [--clustered]                     # Run TorqueBox in clustered mode
      [--data-directory=DATA-DIRECTORY]  # Override the directory TorqueBox
 uses to store it runtime data
  -e, [--extra=EXTRA]                   # Extra options to pass through to
 JBoss AS, you will to escape dashes with \ (e.g. \--help)
      [--max-threads=N]                 # Maximum number of HTTP threads
  -b, [--bind-address=BIND-ADDRESS]      # IP address to bind to - don't set
 this to 0.0.0.0 if used with --clustered
      [--node-name=NODE-NAME]            # Override the name of the node (which
 by default is the hostname)
      [--port-offset=N]                 # Offset all port numbers listened on
 by TorqueBox by this number
  -J, [--jvm-options=JVM-OPTIONS]        # Pass options on to the JVM

Run TorqueBox (binds to localhost, use -b to override)
```

5. Shortcuts For Accessing Paths Inside torquebox-server Gem

With our zip distribution, you set `$TORQUEBOX_HOME`, `$JBOSS_HOME`, and `$JRUBY_HOME`. These aren't set when installing TorqueBox as a gem but we provide an easy way to access those same paths if needed:

```
$ torquebox env torquebox_home
```

The available environment variables are torquebox_home, jboss_home, and jruby_home. Note that they are case-insensitive so you can use TORQUEBOX_HOME if you prefer.

Example 20.1. Tailing AS7 boot.log File

```
$ tail `torquebox env jboss_home`/standalone/log/boot.log
11:26:32,107 INFO  [jacorb.poa] POA RootPOA destroyed
11:26:32,109 INFO  [jacorb.orb] prepare ORB for shutdown...
11:26:32,110 INFO  [jacorb.orb] ORB going down...
11:26:32,112 INFO  [jacorb.orb] ORB shutdown complete
11:26:32,113 INFO  [jacorb.orb.iiop] Listener exited
11:26:32,113 INFO  [jacorb.orb] ORB run, exit
11:26:32,143 INFO  [org.hornetq.core.server.impl.HornetQServerImpl] HornetQ
 Server version 2.2.7.Final (HQ_2_2_7_FINAL_AS7, 121) [612e2de5-f41d-11e0-
b7b8-005056c00008] stopped
11:26:33,782 WARN  [org.torquebox.core.runtime] No initializer set for runtime
11:26:33,801 INFO  [org.torquebox.core.runtime] Created ruby runtime
 (ruby_version: RUBY1_8, compile_mode: JIT, context: global) in 9.86s
11:26:33,806 INFO  [org.jboss.as] JBoss AS 7.0.2.Final "Arc" stopped in 1729ms
```

TorqueBox Production Tips

1. Clustering

1.1. Enabling Clustering

```
$ torquebox run --clustered
```

If you're starting JBoss AS7 directly via standalone.sh, you'll need to pass the server-config option to enable clustering.

```
$ $JBOSS_HOME/bin/standalone.sh --server-config=standalone-ha.xml
```

The --clustered option to torquebox run just chooses the standalone-ha.xml configuration for you under the covers. So, if you need to edit any of the underlying AS7 configuration the file's location is `$JBOSS_HOME/standalone/configuration/standalone-ha.xml`.

In either case, you'll know TorqueBox is running in clustered mode when you see something like the output below in the console upon startup.

```
10:38:17,118 INFO  [stdout] (ServerService Thread Pool -- 86)
10:38:17,118 INFO  [stdout] (ServerService Thread Pool -- 86)
 -------------------------------------------------------------------
10:38:17,118 INFO  [stdout] (ServerService Thread Pool -- 86) GMS:
 address=node2/web, cluster=web, physical address=192.168.1.163:55300
10:38:17,119 INFO  [stdout] (ServerService Thread Pool -- 86)
 --------------------------------------------------------------------
```

When additional nodes are started and become connected to the other nodes, you will seem something like the following in the console of both nodes:

```
10:38:17,226 INFO  [org.infinispan.remoting.transport.jgroups.JGroupsTransport]
 (Incoming-1,null) ISPN000094: Received new cluster view: [node1/web|1] [node1/
web, node2/web]
10:38:18,362 INFO  [org.hornetq.core.server.cluster.impl.BridgeImpl] (Thread-7
 (HornetQ-server-HornetQServerImpl::serverUUID=e40c150a-7d0d-11e2-81a7-
c54946823213-1095366819)) Bridge ClusterConnectionBridge@2d9efd57 [name=sf.my-
cluster.41849c5b-7d0e-11e2-b6fc-f37690770a10, queue=QueueImpl[name=sf.my-
cluster.41849c5b-7d0e-11e2-b6fc-f37690770a10, postOffice=PostOfficeImpl
 [server=HornetQServerImpl::serverUUID=e40c150a-7d0d-11e2-81a7-
c54946823213]]@210a7227 targetConnector=ServerLocatorImpl (identity=(Cluster-
connection-bridge::ClusterConnectionBridge@2d9efd57 [name=sf.my-
```

```
cluster.41849c5b-7d0e-11e2-b6fc-f37690770a10, queue=QueueImpl[name=sf.my-
cluster.41849c5b-7d0e-11e2-b6fc-f37690770a10, postOffice=PostOfficeImpl
 [server=HornetQServerImpl::serverUUID=e40c150a-7d0d-11e2-81a7-
c54946823213]]@210a7227 targetConnector=ServerLocatorImpl
 [initialConnectors=[org-hornetq-core-remoting-impl-netty-
NettyConnectorFactory?port=5545&host=192-168-1-163],
 discoveryGroupConfiguration=null]]::ClusterConnectionImpl@1368605238
 [nodeUUID=e40c150a-7d0d-11e2-81a7-c54946823213, connector=org-hornetq-core-
remoting-impl-netty-NettyConnectorFactory?port=5445&host=192-168-1-163,
 address=jms, server=HornetQServerImpl::serverUUID=e40c150a-7d0d-11e2-81a7-
c54946823213])) [initialConnectors=[org-hornetq-core-remoting-
impl-netty-NettyConnectorFactory?port=5545&host=192-168-1-163],
 discoveryGroupConfiguration=null]] is connected
```

This indicates that the two nodes have successfully connected as part of the cluster.

1.2. Multicast Out of the Box

Clustering is designed to use multicast out of the box. If you're on a network that can't use multicast, see Section 2, “Clustering TorqueBox Without Multicast”

1.3. Don't Bind to 0.0.0.0

JGroups, the underlying library used for most of TorqueBox clustering, doesn't support clustering if bound to 0.0.0.0. Make sure you bind TorqueBox to a real IP address that's accessible from other nodes in the cluster.

2. Clustering TorqueBox Without Multicast

By default when you start TorqueBox in clustered mode other members of the cluster are discovered using multicast. Sometimes this isn't the desired behavior, either because the environment doesn't support multicast or the administrator wants direct control over the members of a cluster. In these cases, it's possible to configure TorqueBox to use a predefined set of cluster members.

Under the hood TorqueBox uses a library called JGroups to handle the cluster discovery and transports. An example of configuring TorqueBox services to cluster without multicast is below.

Example 21.1. JGroups Configuration (`$JBOSS_HOME/standalone/configuration/standalone-ha.xml`)

```
<server xmlns="urn:jboss:domain:1.4">
  <profile>
    ...
    <subsystem xmlns="urn:jboss:domain:jgroups:1.1" default-stack="tcp">
      <stack name="tcp">
        <transport type="TCP" socket-binding="jgroups-tcp">
```

```
          <property name="max_bundle_size">32k</property>
        </transport>
        <protocol type="TCPPING">
          <property name="initial_hosts">
            10.100.10.2[7600],10.100.10.3[7600]
          </property>
        </protocol>
        <protocol type="MERGE3"/>
        <protocol type="FD_SOCK" socket-binding="jgroups-tcp-fd"/>
        <protocol type="FD"/>
        <protocol type="VERIFY_SUSPECT"/>
        <protocol type="pbcast.NAKACK2"/>
        <protocol type="UNICAST3"/>
        <protocol type="pbcast.STABLE"/>
        <protocol type="pbcast.GMS"/>
        <protocol type="UFC"/>
        <protocol type="MFC"/>
        <protocol type="FRAG2">
          <property name="frag_size">30k</property>
        </protocol>
        <protocol type="RSVP"/>
      </stack>
    </subsystem>
    ...
  </profile>
   <socket-binding-group name="standard-sockets" default-interface="public" port-
offset="${jboss.socket.binding.port-offset:0}">
    ...
    <socket-binding name="jgroups-tcp" port="7600"/>
    <socket-binding name="jgroups-tcp-fd" port="57600"/>
    ...
  </socket-binding-group>
</server>
```

Important Changes

The most important changes here are a) replacement of the MPING protocol with the TCPPING protocol with its initial_hosts property and b) the default-stack="tcp" attribute and value added to the <subsystem>. Be sure to replace the initial_hosts IP addresses with the correct values for your environment and change the ports from 7600 if you've changed the jgroups-tcp socket binding to a different port on those hosts.

2.1. Clustering On Amazon EC2

A gossip router is the typical solution when dynamic peer discovery is desired in a non-multicast environment. Another option, if on Amazon EC2, is the S3_PING [http://www.jgroups.org/javadoc/org/jgroups/protocols/S3_PING.html] JGroups protocol.

Enabling clustering with dynamic discovery on EC2 amounts to replacing the `MPING` protocol element of the "tcp" stack configured in `$JBOSS_HOME/standalone/configuration/standalone-ha.xml` with `S3_PING`. And be sure to change the `default-stack` attribute of the `subsystem` to "tcp".

Example 21.2. JGroups Configuration (`$JBOSS_HOME/standalone/configuration/standalone-ha.xml`)

```
...
<subsystem xmlns="urn:jboss:domain:jgroups:1.1" default-stack="tcp">
  <stack name="tcp">
    <transport type="TCP" socket-binding="jgroups-tcp">
      <property name="max_bundle_size">32k</property>
    </transport>

    <protocol type="S3_PING">
      <property name="secret_access_key">YOUR_SECRET_ACCESS_KEY</property>
      <property name="access_key">YOUR_ACCESS_KEY</property>
      <property name="location">SOME_BUCKET_PATH</property>
    </protocol>

    <protocol type="MERGE3"/>
    <protocol type="FD_SOCK" socket-binding="jgroups-tcp-fd"/>
    ...
  </stack>
</subsystem>
```

2.2. HornetQ Configuration

Without multicast, you must change the HornetQ config to use the JGroups "tcp" stack instead of the default "udp" stack.

Search for `jgroups-stack` in `$JBOSS_HOME/standalone/configuration/standalone-ha.xml`, and you'll see this beneath both the `broadcast-group` and `discovery-group` elements:

```
<jgroups-stack>${msg.jgroups.stack:udp}</jgroups-stack>
```

This `${property:default}` syntax refers to a Java system property called `msg.jgroups.stack`. If unset, the value following the colon is used, so you must either set this system property to "tcp" on the command line, e.g. `-Dmsg.jgroups.stack=tcp`, or replace "udp" with "tcp" in the config file for both the `broadcast-group` and `discovery-group` elements.

3. Sizing Number of HTTP Threads to Connection Pool

When running under load in production and against a database, you'll want to size the number of HTTP threads concurrently processing web requests based on the number of connections available in your database connection pool so you don't have too many requests waiting to grab a connection from the pool and timing out. The specific ratio of HTTP threads to database connection pool size will depend on your application, but a good starting point is 1 to 1.

3.1. Setting Database Connection Pool Size

Example 21.3. Database Connection Pool (`config/database.yml`)

```
production:
  adapter: mysql
  database: my_database
  host: my_host
  username: my_username
  password: my_password
  encoding: utf8
  pool: 100
```

This example sets the database connection pool size to 100.

3.2. Setting Max Number of HTTP Threads

If using the `torquebox-server` gem, you can pass the `--max-threads` parameter to set the maximum number of HTTP threads.

```
$ torquebox-server run --max-threads=25
```

If not using the `torquebox-server` gem, you can control the maximum number of HTTP threads by setting a system property.

Table 21.1. Number of HTTP Threads System Property

System Property	Description
`org.torquebox.web.http.maxThreads`	The maximum number of threads to use for the default HTTP connector. If you've changed the connector's name from http in `standalone.xml` then substitute http for the new connector name in the property key.

System Property	Description
	The default value is inherited from AS7 and is 512 * the number of CPUs.

Example 21.4. Number of HTTP Threads (`$JBOSS_HOME/standalone/configuration/standalone.xml`)

```
<extensions>
  ...
</extensions>
<system-properties>
  <property name='org.torquebox.web.http.maxThreads' value='100'/>
</system-properties>
```

This example sets the maximum of HTTP threads to 100.

4. SSL Configuration

4.1. SSL Termination at Load Balancer

If you choose to terminate SSL at the load balancer, you'll want to set the request header X_FORWARDED_PROTO to 'https' before forwarding the request to TorqueBox. Rails will pick up on this header automatically but other web frameworks may require you to check this header manually to determine if a request came in over HTTP or HTTPS.

To set this header under Apache, add the following line to the HTTPS VirtualHost configuration:

```
set X_FORWARDED_PROTO 'https'
```

4.2. SSL Termination at TorqueBox

Another option is to terminate SSL connections at TorqueBox. This requires editing the appropriate configuration file - `$JBOSS_HOME/standalone/configuration/standalone.xml` when not running in a cluster or `$JBOSS_HOME/standalone/configuration/standalone-ha.xml` when running in a cluster.

Example 21.5. SSL Configuration in standalone.xml

```
<subsystem xmlns="urn:jboss:domain:web:1.4" native="false" default-virtual-server="default-host"
  <connector name="http" protocol="HTTP/1.1" scheme="http" socket-binding="http"/
```

```
 <connector name="https" protocol="HTTP/1.1" scheme="https" socket-binding="https"
 secure="true"
   <ssl name="https" key-alias="myalias" password="foobar" certificate-key-file="/
tmp/keystore"/
  </connector
  <virtual-server name="default-host"
    <alias name="localhost"/
    <alias name="example.com"/
  </virtual-server
</subsystem
```

This is an example of the entire JBoss Web subsystem after being configured to terminate SSL.

5. JVM Tuning

5.1. CodeCache

In a production system it's very important to not let the JVM CodeCache get full. The CodeCache is an area of memory where the bytecode from all JITted methods gets stored. If it fills up, methods will no longer be JITted and things will run slower than they could. The default CodeCache size varies by JVM and platform, but for most servers it's only 48MB.

When you run out of CodeCache space, a warning will be logged in the TorqueBox `server.log` that looks like this:

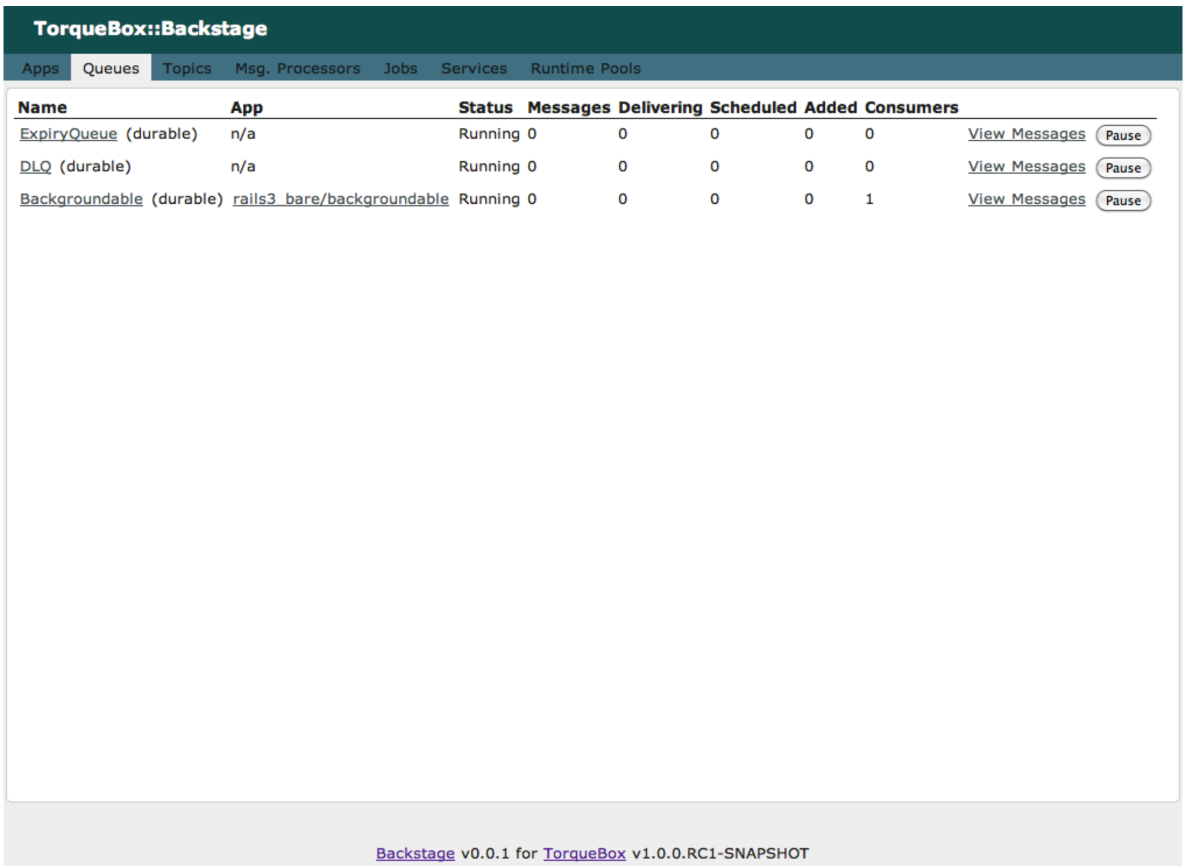
```
Java HotSpot(TM) 64-Bit Server VM warning: CodeCache is full. Compiler has been
 disabled
```

To increase the CodeCache size, you'll want to set the JVM property `-XX:ReservedCodeCacheSize=256m`, replacing the 256m with the desired size of your CodeCache. See Section 3, “Setting JVM Properties” for details on how to set this JVM property.

TorqueBox Additional Resources

1. BackStage

BackStage is a Sinatra app that you may deploy within TorqueBox to get additional views and control into your application's components.



1.1. Features

Applications. View all deployed Ruby applications.

Destinations. Enumerate and interrogate messaging queues and topics. Allows browsing of messages within queues.

Message Processors. Control message processors, including pausing their execution.

Scheduled Jobs. Scheduled jobs can be paused.

Ruby Runtime Pools. View information about the runtime pools for all applications. Allows arbitrary script execution within a runtime from a pool.

1.2. More Information

More Information About Backstage May Be Found On The Torquebox Website. The Source For Backstage Is Hosted At Github.

- Http://torquebox.org/backstage [http://torquebox.org/backstage]
- Http://github.com/torquebox/backstage [http://github.com/torquebox/backstage]

2. New Relic

New Relic [http://newrelic.com] is an application monitoring and management tool that provides statistics about and insight into your application. New Relic will work with TorqueBox just like any other Ruby application. There are some minor caveats regarding JRuby which New Relic lists on their FAQ [https://newrelic.com/docs/general/new-relic-on-jruby].

2.1. Usage

Using New Relic with TorqueBox is just like using it with any other Ruby application. You simply install the gem and ensure it's available to your application. This is usually accomplished using Bundler. Additionally, TorqueBox provides some basic integration with New Relic's Background Tasks tab. Methods run in background tasks via `always_background` or `obj.background.some_method` will appear in this tab. Complete details about New Relic's Ruby integration can be found on the New Relic website [https://docs.newrelic.com/docs/ruby/new-relic-on-jruby].

Note: when using New Relic with Rails 2.3.x you may need to add the following code to your `config/environments/production.rb` file.

```
begin
  require 'newrelic_rpm'
  NewRelic::Agent
rescue
  # Log the exception, mayhap
end
```

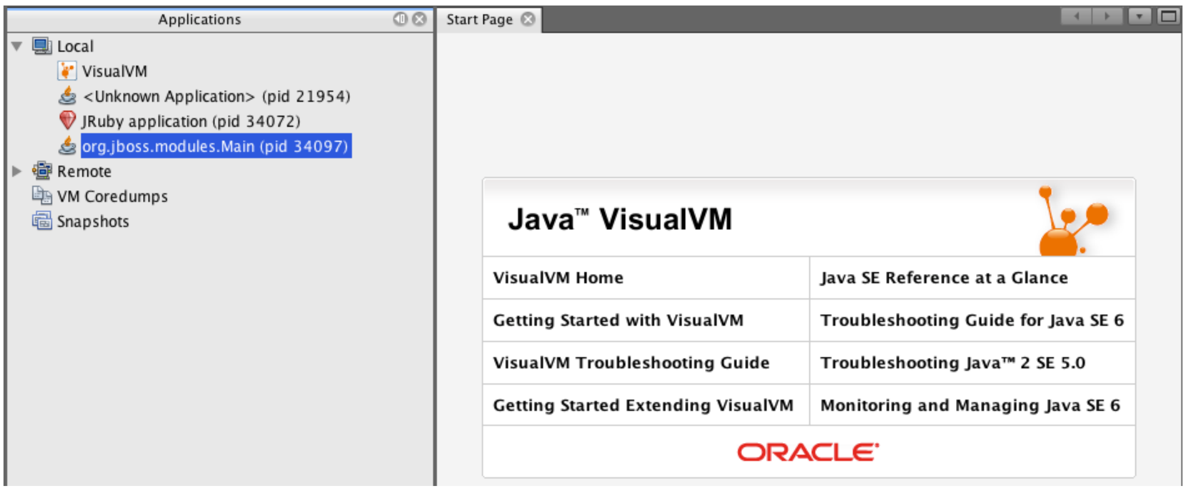
3. VisualVM

VisualVM is a useful tool for monitoring and troubleshooting Java applications. Detailed below are the steps to connect to TorqueBox from VisualVM when connecting to both local and remote TorqueBox servers. More information about VisualVM itself can be found in the Oracle VisualVM documentation [http://docs.oracle.com/javase/6/docs/technotes/guides/visualvm/index.html]. To start VisualVM from the command line:

```
$jvisualvm
```

3.1. Connecting VisualVM to a Local TorqueBox

With TorqueBox running locally, simply start VisualVM and connect to the "org.jboss.modules.Main" application listed.



3.2. Connecting VisualVM to a Remote TorqueBox

Connecting to a remote TorqueBox server requires a few more steps.

On the remote TorqueBox server, edit `$JBOSS_HOME/standalone/configuration/standalone.xml` or, if running in a cluster, `$JBOSS_HOME/standalone/configuration/standalone-ha.xml` and instruct the JMX subsystem to not use the management endpoint and instead use the remoting endpoint for remote JMX connections.

Example 22.1. JMX Configuration in standalone.xml

```
<subsystem xmlns='urn:jboss:domain:jmx:1.2'>
  <expose-resolved-model/>
  <expose-expression-model/>
  <remoting-connector use-management-endpoint='false'/>
</subsystem>
```

Next we need to add a new application user to the remote TorqueBox server.

```
$ $JBOSS_HOME/bin/add-user.sh
```

```
What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): b

Enter the details of the new user to add.
Realm (ApplicationRealm) :
Username : testuser
Password :
Re-enter Password :
What roles do you want this user to belong to? (Please enter a comma separated
 list, or leave blank for none) :
About to add user 'testuser' for realm 'ApplicationRealm'
Is this correct yes/no? yes
Added user 'testuser' to file '/opt/torquebox/jboss/standalone/configuration/
application-users.properties'
Added user 'testuser' to file '/opt/torquebox/jboss/domain/configuration/
application-users.properties'
Added user 'testuser' with roles  to file '/opt/torquebox/jboss/standalone/
configuration/application-roles.properties'
Added user 'testuser' with roles  to file '/opt/torquebox/jboss/domain/
configuration/application-roles.properties'
Is this new user going to be used for one AS process to connect to another AS
 process?
e.g. for a slave host controller connecting to the master or for a Remoting
 connection for server to server EJB calls.
yes/no? yes
To represent the user add the following to the server-identities definition
 <secret value="cDRzc3cwcmQj" /
```

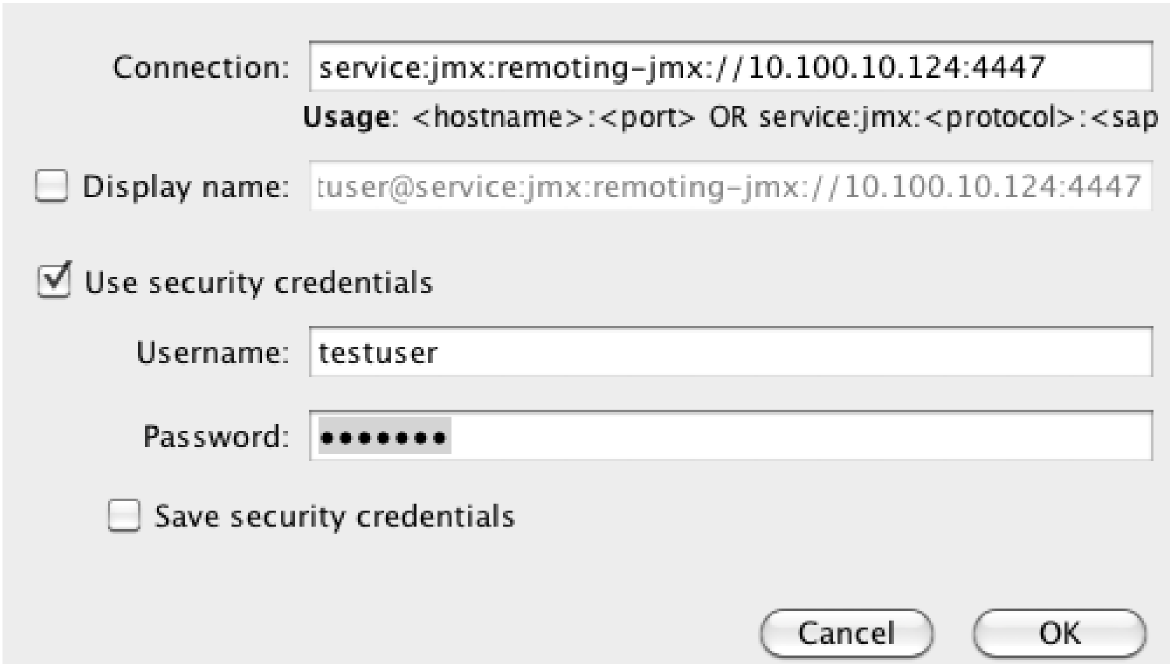
Finally, start TorqueBox on the remote server and bind it to a real IP address.

```
$ torquebox run -b <server_ip>
```

When running VisualVM on the local machine you'll need to ensure `$JBOSS_HOME/bin/client/jboss-cli-client.jar` is on the classpath. You can simply copy that file to the local machine or if TorqueBox is installed locally use the jar from inside there.

```
$ jvisualvm --cp:a $JBOSS_HOME/bin/client/jboss-cli-client.jar
```

Inside VisualVM, click on File -> Add JMX Connection. The connection string is "service:jmx:remoting-jmx://<remote_server_ip>:4447". Also check "Use Security Credentials" and enter the username / password used in the add-user.sh script.


Connection: service:jmx:remoting-jmx://10.100.10.124:4447
Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap
Display name: tuser@service:jmx:remoting-jmx://10.100.10.124:4447
Use security credentials
Username: testuser
Password:
Save security credentials
Cancel
OK

Appendix A. Licensing

A variety of third-party components are used in the construction of TorqueBox.

JBoss AS & TorqueBox.

JBoss AS and TorqueBox are licensed under the LGPL (see Appendix B, GNU Lesser General Public License version 3). The TorqueBox documentation is licensed under a Creative-Commons license (see Appendix N, Creative Commons Attribution-ShareAlike 3.0).

JRuby and JRuby-Rack.

Portions of code have been borrowed from JRuby (see Appendix C, JRuby Licenses) and JRuby-Rack (see Appendix D, JRuby-Rack License).

Appendix B. GNU Lesser General Public License version 3

JBoss AS and TorqueBox License

Version 3, 29 June 2007

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

a. under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

b. under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

a. Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.

b. Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a. Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

b. Accompany the Combined Work with a copy of the GNU GPL and this license document.

c. For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

d. Do one of the following:

    1. Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

    2. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e. Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

b. Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

Appendix C. JRuby Licenses

JRuby is Copyright (c) 2007-2010 The JRuby project, and is released under a tri CPL/GPL/LGPL license. You can use it, redistribute it and/or modify it under the terms of the:

- Common Public License version 1.0
- GNU General Public License version 2
- GNU Lesser General Public License version 2.1

`lib/ruby/1.9/drb/*`, `build_lib/bytelist.jar` (`http://github.com/jruby/bytelist`), `build_lib/jruby-embed.jar` (`http://kenai.com/projects/jruby-embed`) and `build_lib/yydebug.jar` (`http://svn.codehaus.org/jruby/trunk/jay/yydebug`) are released under the same copyright/license.

Some additional libraries distributed with JRuby are not covered by JRuby's licence. Most of these libraries and their licenses are listed below. Also see LICENSE.RUBY for most files found in src/lib/ruby/1.8.

`bench/rails/public/javascripts/*` are distributed under the MIT license, and have the following copyrights:

```
controls.js is Copyright:
(c) 2005-2008 Thomas Fuchs (http://script.aculo.us, http://mir.aculo.us)
(c) 2005-2007 Ivan Krstic (http://blogs.law.harvard.edu/ivan)
(c) 2005-2007 Jon Tirsen (http://www.tirsen.com)

dragdrop.js is Copyright:
(c) 2005-2008 Thomas Fuchs (http://script.aculo.us, http://mir.aculo.us)
(c) 2005-2007 Sammi Williams (http://www.oriontransfer.co.nz, sammi@oriontransfer.co.nz)

effect.js is Copyright (c) 2005-2008 Thomas Fuchs.

prototype.js is Copyright (c) 2005-2007 Sam Stephenson.
```

`build_lib/*asm*jar` (`http://asm.objectweb.org`) are distributed under the BSD license.

`build_lib/apt-mirror-api.jar`, `build_lib/bnd-0.0.249.jar`, `build_lib/commons-logging-1.1.1.jar`, `build_lib/joda-time-1.5.1.jar`, BSF and ant are distributed under the Apache Software License, Version 1.1 (license file inside the jars).

`build_lib/constantine.jar` (`http://kenai.com/projects/constantine`), `build_lib/jcodings.jar` (`http://github.com/jruby/jcodings`), `build_lib/jaffl.jar` (`http://projectkenai.com/projects/jaffl`) and `build_lib/joni.jar` (`http://github.com/jruby/joni`) are distributed under the MIT license.

`build_lib/jarjar-1.0rc8.jar` (`http://code.google.com/p/jarjar`), `build_lib/joda-time-1.5.1.jar` (`http://joda-time.sourceforge.net`), `build_lib/dynalang-0.3.jar` (`http://dynalang.sourceforge.net`), `build_lib/nailgun-0.7.1.jar` and `tool/nailgun/ng.exe` (`http://martiansoftware.com/nailgun`) are distributed under the Apache License version 2.0.

`build_lib/emma*jar` (`http://emma.sourceforge.net`) and `build_lib/junit.jar` (`http://www.junit.org`) are distributed under the Common Public License v1.0.

`build_lib/jffi*jar` (`http://kenai.com/projects/jffi`) and `build_lib/jgrapht-jdk1.5.jar` (`http://jgrapht.sourceforge.net`) are distributed under the GPL v3.

`build_lib/jline-0.9.93` (`http://jline.sourceforge.net`) is distributed under the following license:

```
Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
```

```
conditions are met:

Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with
the distribution.

Neither the name of JLine nor the names of its contributors
may be used to endorse or promote products derived from this
software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
```

`build_lib/jna*jar` (`http://jna.dev.java.net`) are distributed under the LGPL-2.1+ license.

`build_lib/invokedynamic.jar` and `build_lib/jsr292-mock.jar` (`http://code.google.com/p/jvm-language-runtime`) are distributed under the LGPL license.

Bouncycastle is released under the MIT license, and is Copyright (c) 2000 - 2006 The Legion Of The Bouncy Castle.

Jyvaml and rbyaml are released under the MIT license, and are Copyright (c) 2006 Ola Bini.

Appendix D. JRuby-Rack License

Appendix E. LICENSE AGREEMENT AND LIMITED PRODUCT WARRANTY

LIBERATION FONT SOFTWARE

This agreement governs the use of the Software and any updates to the Software, regardless of the delivery mechanism. Subject to the following terms, Red Hat, Inc. ("Red Hat") grants to the user ("Client") a license to this work pursuant to the GNU General Public License v.2 with the exceptions set forth below and such other terms as are set forth in this End User License Agreement.

1. The Software and License Exception. LIBERATION font software (the "Software") consists of TrueType-OpenType formatted font software for rendering LIBERATION typefaces in sans-serif, serif, and monospaced character styles. You are licensed to use, modify, copy, and distribute the Software pursuant to the GNU General Public License v.2 with the following exceptions:

(a) As a special exception, if you create a document which uses this font, and embed this font or unaltered portions of this font into the document, this font does not by itself cause the resulting document to be covered by the GNU General Public License. This exception does not however invalidate any other reasons why the document might be covered by the GNU General Public License. If you modify this font, you may extend this exception to your version of the font, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

(b) As a further exception, any distribution of the object code of the Software in a physical product must provide you the right to access and modify the source code for the Software and to reinstall that modified version of the Software in object code form on the same physical product on which you received it.

2. Intellectual Property Rights. The Software and each of its components, including the source code, documentation, appearance, structure and organization are owned by Red Hat and others and are protected under copyright and other laws. Title to the Software and any component, or to any copy, modification, or merged portion shall remain with the aforementioned, subject to the applicable license. The "LIBERATION" trademark is a trademark of Red Hat, Inc. in the U.S. and other countries. This agreement does not permit Client to distribute modified versions of the Software using Red Hat's trademarks. If Client makes a redistribution of a modified version of the Software, then Client must modify the files names to remove any reference to the Red Hat trademarks and must not use the Red Hat trademarks in any way to reference or promote the modified Software.

3. Limited Warranty. To the maximum extent permitted under applicable law, the Software is provided and licensed "as is" without warranty of any kind, expressed or implied, including the implied warranties of merchantability, non-infringement or fitness for a particular purpose. Red Hat does not warrant that the functions contained in the Software will meet Client's requirements or that the operation of the Software will be entirely error free or appear precisely as described in the accompanying documentation.

4. Limitation of Remedies and Liability. To the maximum extent permitted by applicable law, Red Hat or any Red Hat authorized dealer will not be liable to Client for any incidental or consequential damages, including lost profits or lost savings arising out of the use or inability to use the Software, even if Red Hat or such dealer has been advised of the possibility of such damages.

5. General. If any provision of this agreement is held to be unenforceable, that shall not affect the enforceability of the remaining provisions. This agreement shall be governed by the laws of the State of North Carolina and of the United States, without regard to any conflict of laws provisions, except that the United Nations Convention on the International Sale of Goods shall not apply.

Appendix F. GNU General Public License

Version 2, June 1991

Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor,
Boston, MA 02110-1301
USA

Version 2, June 1991

1. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and

2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

2. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

2.1. Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2.2. Section 1

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2.3. Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

2.4. Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

2.5. Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

2.6. Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

2.7. Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

2.8. Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

2.9. Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

2.10. Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

2.11. Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

2.12. NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

2.13. Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

3. How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type “show w”. This is free software, and you are welcome to redistribute it under certain conditions; type “show c” for details.

The hypothetical commands “show w” and “show c” should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than “show w” and “show c”; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program “Gnomovision” (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix G. GNU General Public License version 3

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users’ Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
one line to give the program’s name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program.  If not, see http://www.gnu.org/licenses/.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
program Copyright (C) year name of author
This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘`show w`’ and ‘`show c`’ should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

Appendix H. GNU Lesser General Public License

This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.

Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor,
Boston,
MA
02110-1301
USA

Version 2.1, February 1999

1. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method:

1. we copyright the library, and

2. we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the Lesser General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a “work based on the library” and a “work that uses the library”. The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

2. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

2.1. Section 0

This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called “this License”). Each licensee is addressed as “you”.

A “library” means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The “Library”, below, refers to any such software library or work which has been distributed under these terms. A “work based on the Library” means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term “modification”.)

“Source code” for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

2.2. Section 1

You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2.3. Section 2

You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. The modified work must itself be a software library.

b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

   (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

2.4. Section 3

You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

2.5. Section 4

You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

2.6. Section 5

A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

2.7. Section 6

As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

2.8. Section 7

You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

2.9. Section 8

You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

2.10. Section 9

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

2.11. Section 10

Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

2.12. Section 11

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

2.13. Section 12

If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

2.14. Section 13

The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

2.15. Section 14

If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

2.16. NO WARRANTY Section 15

BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

2.17. Section 16

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

3. How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990 Ty Coon, President of Vice

That's all there is to it!

Appendix I. Apache Software License, Version 1.1

Appendix J. Apache License, Version 2.0

Apache License

Version 2.0, January

2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

1. APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner] Licensed under the Apache License, Version
2.0 (the "License"); you may not use this file except in compliance with the License.
You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
```

Appendix K. Common Public License Version 1.0 (CPL)

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

> a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
>
> b) in the case of each subsequent Contributor:
>
> i) changes to the Program, and
>
> ii) additions to the Program;
>
> where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

> a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.
>
> b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.
>
> c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.
>
> d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

a) it complies with the terms and conditions of this Agreement; and

b) its license agreement:

i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and

iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

a) it must be made available under this Agreement; and

b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

Appendix L. BSD License

Copyright (c) The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Appendix M. MIT License

Appendix N. Creative Commons Attribution-ShareAlike 3.0

TorqueBox Documentation License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

1. "Adaptation" means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

2. "Collection" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined below) for the purposes of this License.

3. "Creative Commons Compatible License" means a license that is listed at http://creativecommons.org/compatiblelicenses that has been approved by Creative Commons as being essentially equivalent to this License, including, at a minimum, because that license: (i) contains terms that have the same purpose, meaning and effect as the License Elements of this License; and, (ii) explicitly permits the relicensing of adaptations of works made available under that license under this License or a Creative Commons jurisdiction license with the same License Elements as this License.

4. "Distribute" means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.

5. "License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.

6. "Licensor" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

7. "Original Author" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

8. "Work" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a

compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

9. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

10. "Publicly Perform" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

11. "Reproduce" means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

1. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;

2. to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";

3. to Distribute and Publicly Perform the Work including as incorporated in Collections; and,

4. to Distribute and Publicly Perform Adaptations.

5. For the avoidance of doubt:

   a. Non-waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

   b. Waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,

   c. Voluntary License Schemes. The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

1. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(c), as requested.

2. You may Distribute or Publicly Perform an Adaptation only under the terms of: (i) this License; (ii) a later version of this License with the same License Elements as this License; (iii) a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g., Attribution-ShareAlike 3.0 US)); (iv) a Creative Commons Compatible License. If you license the Adaptation under one of the licenses mentioned in (iv), you must comply with the terms of that license. If you license the Adaptation under the terms of any of the licenses mentioned in (i), (ii) or (iii) (the "Applicable License"), you must comply with the terms of the Applicable License generally and the following provisions: (I) You must include a copy of, or the URI for, the Applicable License with every copy of each Adaptation You Distribute or Publicly Perform; (II) You may not offer or impose any terms on the Adaptation that restrict the terms of the Applicable License or the ability of the recipient of the Adaptation to exercise the rights granted to that recipient under the terms of the Applicable License; (III) You must keep intact all notices that refer to the Applicable License and to the disclaimer of warranties with every copy of the Work as included in the Adaptation You Distribute or Publicly Perform; (IV) when You Distribute or Publicly Perform the Adaptation, You may not impose any effective technological measures on the Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.

3. If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv) , consistent with Ssection 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

4. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

1. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

2. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.